eAuthorization Architecture and Platform Implementation

Dhouha Ayed (Thales), Stefan Thaler (TUE), Jerry den Hartog (TUE), Dieter Sommer(IBM), Cyrille Martins (Thales), Daniel Kovacs (IBM), Dennis Gessner (NEC)

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no: 611659

Project Information

Authentication and Authorization for Entrusted Unions

Project number:	611659
Strategic objective:	ICT-2013.1.4.e
Starting date:	2013-12-01
Ending data:	2015-11-30
Website:	http://au2eu.eu/

Document Information

Title: < eAuthorization Architecture and Platform Implementation >

ID: <D3.3.1> Type: P Disemination level: PU

Month: M12 Release date: 30.Nov.2014

Deliverable Description

This document describes the AU2EU eAuthorization framework architecture, and provides the implementation of its components;

Contributors, Editor & Reviewer Information

Contributors (person(partner): sections)	Dhouha Ayed(THA): First Section, Second Section, Third Section, Fourth Section, Ninth Section, Eleventh Section Stefan Thaler, Jerry Den Hartog (TUE): Fifth Section, Seventh Section Cyrille Martins, Dhouha Ayed: Sixth section Dieter Sommer (IBM): Eighth section Dennis Gessner (THA): Tenth Section
Editor (person/partner)	Dhouha Ayed/Thales
Reviewer (person/partner)	Paul Kauster (PRE) Dieter Sommer (IBM)

Release History

Release number	Date issued	Milestone*	SVN version	Release description / changes made
1	11 August 2014	PCOS		Planned contents of different sections
2	18 October 2014	Proposed		First version ready for internal review
3	28 November	Revised		Internal reviews taken into account
4	1 December 2014	Approved, Released		Version released to the Commission Services
5	7 January 2015	Released		Revised version based on project reviewers comments.

* The project uses a multi-stage internal review and release process, with defined milestones. Milestone names include abbreviations/terms as follows:

1. PCOS = ”Planned Content and Structure” (describes planned contents of different sections)
2. Intermediate: Document is approximately 50% complete – review checkpoint
3. External: For release to commission and reviewers;
4. proposed: Document authors submit for internal review
5. revised: Document authors produce new version in response to internal reviewer comments
6. approved: Internal project reviewers accept the document
7. released: Project Technical Manager/Coordinator release to Commission Services

AU2EU Consortium

Full Name	Abbreviated Name	Country
Technische Universiteit Eindhoven	TU/e	Netherlands
Philips Electronics Nederland B.V.	PRE	Netherlands
Bicore Services B.V.	BIC	Netherlands
NEC Europe LTD	NEC	United Kingdom
IBM Research GMBH	IBM	Switzerland
Deutsches Rotes Kreuz	DRK	Germany
Thales Communications & Security SAS	THA	France
Commonwealth Scientific and Industrial Research Organisation	CSO	Australia
Edith Cowan University	ECU	Australia
Royal Melbourne Institute of Technology	RMI	Australia
University of New South Wales	NSW	Australia
Macquarie University	MQ	Australia

Table 1: Consortium Members

Table of Contents

Release History 5
AU2EU Consortium 6
Table of Contents 7
1 Executive Summary 10
2 About this Document 11
2.1 Role of the deliverable 11
2.2 Relationship to other AU2EU deliverables 11
2.3 Structure of this document 11
3 Context and challenges 12
4 Authorization framework architecture 14
4.1 XACML access control components 14
4.2 Authorization mapping components 16
4.2.1 Administration process and tools 16
4.2.2 Runtime ontology conversion 17
4.3 Integrated architecture 18
4.4 Conclusion 22
5 Ontology based conversion service 24
5.1 Ontology 24
5.1.1 Definition of terms 24
5.1.2 Ontology structure 24
5.1.3 Trust in converted attributes 25
5.1.4 Building the ontology 26
5.2 Requirements 26
5.3 Design 27
5.4 REST API 29
5.5 Installation guide 30
5.5.1 Hardware requirements 30
5.5.2 Software requirements 30
5.5.3 Installation process 31
5.6 Examples 31
5.6.1 Example usage – identifying similar concepts 32
5.6.2 Example usage – validating similarity of concepts 32
6 Local policy validation tool 33
6.1 Requirements 33

6.2 Design 33
6.3 Interfaces 34
6.4 Installation guide 35
6.4.1 Hardware requirements 35
6.4.2 Software requirements 35
6.4.3 Installation process 35
6.5 Examples 35
7 Policy conflict resolution tool 40
7.1 Design 40
7.2 Interfaces 41
7.3 Installation guide 42
7.3.1 Hardware requirements 42
7.3.2 Software requirements 42
7.3.3 Installation process 43
8 Authentication/Authorization translation component 44
8.1 Requirements 44
8.2 Design 45
8.3 REST API 46
9 XACML access control framework 48
9.1 Policy Decision Point 48
9.1.1 REST API 48
9.1.2 Installation guide 48
9.1.3 Installation Checking 49
9.1.4 Example 49
10 Mobile authorization platform (NEC) 52
10.1 Requirements 52
10.2 Design 52
10.3 Interfaces 54
10.4 Installation guide 54
10.4.1 Hardware requirements 54
10.4.2 Software requirements 54
10.4.3 Installation process 54
10.4.4 Troubleshooting 55
11 Conclusion 56
Bibliography 58

A. Appendix 59

1 Executive Summary

One key goal of the AU2EU project is to implement and demonstrate an integrated eAuthentication and eAuthorization framework, which enables trusted collaborations and delivery of services across different organizational jurisdictions.

Organizations from different domains collaborate by sharing their services and resources. In order to ensure proper collaboration between them, a shared authorization system is needed. A common problem that occurs when placing such system in place, is how to describe authorization policies and attributes using different concepts and terminologies.

Traditionally authorization is treated as separate concept from authentication, with the only relation between them being that a user needs to be authenticated before she can be authorized to access a resource. Such a separation results in users needing to fully identify themselves by revealing all information before access can occur. Fully revealing information that is not always necessary can compromise user privacy.

This deliverable (D3.3.1) is the third deliverable of WP3. The objective of WP3 is to develop the eAuthorization platform for cross-organizational and jurisdictional distributed systems.

D3.3.1 details the results of Task Task 3.3 “Platform for Advanced Cross-domain Authorization” related to the first period. The objective of this task is to implement the eAuthorization framework for cross-domain collaborative environments by developing administration, decision, and enforcement services for the involved domains as well as services that allow security administrators to match authorizations, to detect conflicts and resolve them. The integration of the eAuthorization framework with a data-minimization authentication system is paramount for a high protection of user attributes.

This document describes the components of the eAuthorization framework, their design and a first implementation. It integrates the results of Task 3.1 (attribute mapping and administration tools) and Task 3.2 (Architecture for integration authorization with eAuthentication). The document gives an integrated overview of the framework components and their roles,. It includes sections on each component, describes design and interfaces, and provides a user guide. The following components are specified: an ontology converter to map authorization attributes, a XACML policy validation tool, a conflict resolution tool, an AuthN/AuthZ mediator to map authentication and authorization attributes, a XACML framework, and a deployment of the authorization framework in a mobile environment.

A second version of this document that describes the final implementation version of the eAuthorization platform will be delivered in the end of the second period.

2 About this Document

2.1 Role of the deliverable

The goal in Task 3.3 is to implement the eAuthorization framework for cross-domain collaborative environments. We can achieve this objective through the design and implementation of administration tools that allow security administrators of various organizations to map authorizations and detect and resolve inconsistencies and conflicts. In addition, decision and enforcement services for the involved domains are needed. This document integrates the results of Task 3.1 (attribute mapping and administration tools) and Task 3.2 (assets related to integration of authentication and authorization) into the eAuthorization framework. It details the components of the eAuthorization platform, their design and a first prototype.

2.2 Relationship to other AU2EU deliverables

This document takes into account the requirements defined in D1.2.1 “Requirements specification document” as well as the use case scenarios specified in D1.1.1 “Detailed descriptions of use cases”. These requirements and use cases drive the technology, which fulfills the authorization gaps in the state of the art, and are therefore the focus of this task.

D1.4 “Reference architecture for eAuthentication & eAuthorization” provides recommendations for the overall architecture of AU2EU, to which the authorization platform must conform. The eAuthorization platform assets are integrated into the eAuthentication platform as specified in the reference architecture described in D1.4.1.

The results of D3.1.1 “Tools for semantic mapping of attributes” and D3.2.1 “Architecture for integrating authorization with authentication” are developed and integrated in this deliverable.

2.3 Structure of this document

This document details the design and the implementation of the eAuthorization framework components for a cross-domain support. Section 3 introduces the context. Section 4 provides a general overview of the eAuthorization framework components. Sections 5, 6, 7, 8 and 9 detail the components of the framework. Section 5 is dedicated to semantic conversion mechanisms in multi-domain contexts. Section 6 is focused on the validation of local access control policies. Section 7 details solutions to resolve conflicts. Section 8 is about policy mapping between the authentication and authorization platform. Section 9 focuses on the standard XACML components provided by the framework. Section 10 is about the extension of the framework for mobile users. Finally, Section 11 concludes the document.

3 Context and challenges

Multiple parties may need to collaborate. A difficult challenge is to conciliate the different affiliations to security domains, the authorities that govern them and the distributed nature of the system. When an initial trust level is established, and a decision to join a collaboration is made, authorization processes pose the first challenge. Beyond the problems related to regulation of access to information, and the guarantee of confidentiality of information, the biggest challenge is the existence of different concepts and strategies to describe access control policies in the various participating organizations. Attributes (roles, identity attributes, resources attributes) that are utilized in access control policies differ from one organization to another, and there is no standard on sharing of access control policies. This problem is illustrated in **Error! Reference source not found.**.

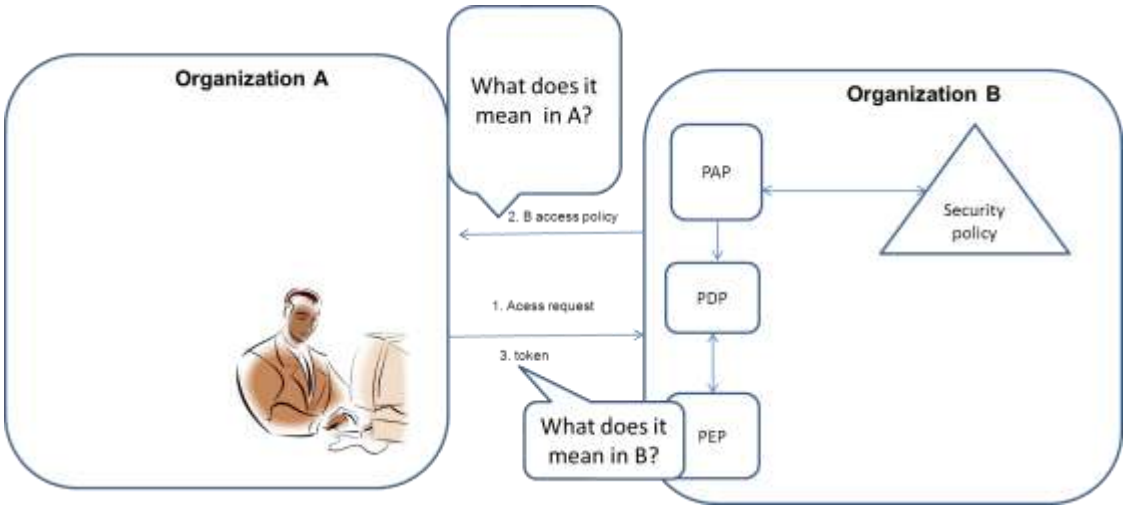

Figure 1 Need of mapping of authorization concepts

Therefore, there is a need for mechanisms that can align different roles and attributes that are locally used in access control policies by organizations, to define equivalences across domains when needed.

Another set of problems that occur when multiple organizations collaborate are that access control policies are not easily understandable, their effect is not always predictable, and conflicts in authorization decision over shared resources may occur due to different interests of the involved parties.

The next challenge occurs after taking a decision to join the collaboration: the authentication process. Traditional authorization systems require all of the users' information that may be relevant to an authorization decision. Consequently, the authentication components need to fully identify the users, and to collect all available information about them. This negates any privacy and security benefits of data-minimization authentication technologies, such as private credential systems. Credential protection is an important requirement of the AU2EU pilots. However, there is a high complexity in the implementation and integration of privacy-preserving authentication techniques into existing authorization systems.

D3.1.1 describes the design and the principle of the mechanisms for mapping authorization attributes in a multi-organizational context. The document describes a mapping solution that is structured on a common base ontology, alignment between domain vocabulary of different organizations to this base ontology, and an ontology conversion service, which dynamically converts authorization attributes of different organizations. In addition, D3.1.1 analyses ontology verification tools, describes a logic-

based policy validation approach for standard XACML access control policies and a solution for dynamic conflict resolution in a multi-organizational context.

D3.2.1 identifies design questions related to architecture that integrates disparate authentication and authorization systems. A number of architectural possibilities are described and analyzed and an architectural framework that enables the integration of a standard XACML authorization chain, with ABC4Trust authentication mechanisms is proposed.

The objective of this deliverable is to define and implement an authorization framework that extends an attribute-based authorization system, such as XACML, as a standard, with components that support authorization attributes mapping and integration with data-minimization authentication system.

The framework definition and implementation is based on the integrated architecture defined in D3.2.1 and the results of D3.1. 1. According to the requirements analysis and the results of these two previous deliverables, the eAuthorization framework provides the following components and functionalities:

- A XACML framework
- An ontology edition and alignment mechanisms providing functionalities to merge and align domain vocabulary of various organizations to the common base ontology
- An ontology conversion service which dynamically converts authorization attributes of different issuers
- Ontology verification mechanisms
- A logical-based XACML validation tool
- A Cross-organization conflict resolution tool
- AuthZ/AuthN mediation service to map authorization and authentication policies

The next section is an overview of these components, and the sections that follow describe the details of each component per section.

4 Authorization framework architecture

This section describes an abstract architecture of the AU2EU authorization framework by giving an overview on each of its components and their role. Section 4.1details the standard XACML components that are provided by the framework. Section 4.2 describes the components that support authorization mapping between various organizations. Section 4.3 describes the components of an extended XACML framework that allow the authorization framework to be integrated with a data-minimization authentication framework and to provide cross-organization dynamic conversion of attributes.

4.1 XACML access control components

XACML (eXtensible Access Control Markup Language) is an OASIS standard [2]which defines a flexible attribute-based framework for access control. The standard defines a language for expressing security policy, based on XML.

The language defines methods to:

- combine individual rules and policies into a single policy set that applies to a particular decision request;
- combine rules and policies;
- deal with multiple subjects acting in different capacities;
- base an authorization decision on attributes of the subject and resource;
- deal with multi-valued attributes;
- provide a set of logical and mathematical operators on attributes of the subject, resource and environment;
- specify a set of actions (i.e. obligations) that must be performed in conjunction with policy enforcement.

The standard introduces a number of components, mainly:

- the policy administration point (PAP), i.e. the system entity that creates a policy or policy set;
- the policy decision point (PDP), i.e. the system entity that evaluates applicable policy and renders an authorization decision;
- the policy enforcement point (PEP), i.e. the system entity that performs access control, by making decision requests and enforcing authorization decisions;
- the policy information point (PIP), i.e. the system entity that acts as a source of attribute values;
- the context handler, i.e. the system entity that converts decision requests in the native request format to the XACML canonical form and converts authorization decisions in the XACML canonical form to the native response format.

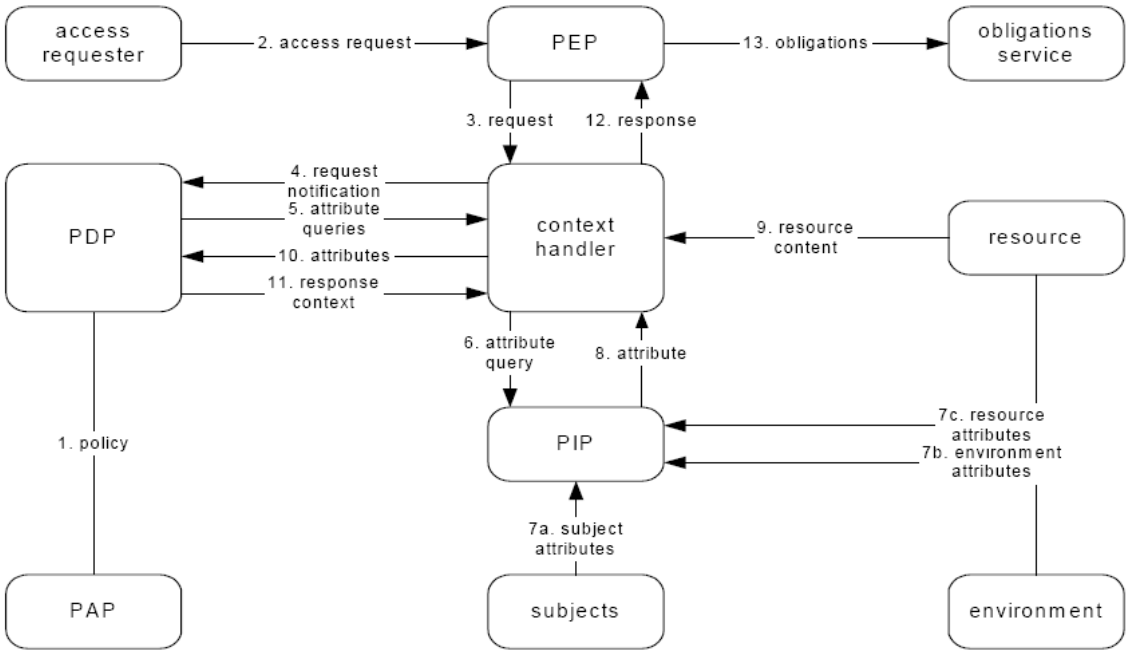

Figure 2: Data flow diagram of XACML

The model operates by the following workflow (see Figure 2):

1. PAPs write policies and policy sets and make them available to the PDP. These policies or policy sets represent the complete policy for a specified target.

2. The access requester sends a request for access to the PEP.

3. The PEP sends the request for access to the context handler in its native request format, optionally including attributes of the subjects, resource, action and environment.

4. The context handler constructs an XACML request context and sends it to the PDP.

5. The PDP requests any additional subject, resource, action and environment attributes from the context handler.

6. The context handler requests the attributes from a PIP.

7. The PIP obtains the requested attributes.

8. The PIP returns the requested attributes to the context handler.

9. Optionally, the context handler includes the resource in the context.

10. The context handler sends the requested attributes and (optionally) the resource to the PDP. The PDP evaluates the policy.

11. The PDP returns the response context (including the authorization decision) to the context handler.

12. The context handler translates the response context to the native response format of the PEP. The context handler returns the response to the PEP.

13. The PEP fulfils the obligations.

14. (Not shown) If access is permitted, then the PEP permits access to the resource; otherwise, it denies access.

Finally, it is important to recall that XACML is a machine-interpretable policy format, not a policy authoring language for humans.

4.2 Authorization mapping components

The mapping of access control attributes is provided through two main mechanisms: the alignment of authorization concepts by domain administrators (offline) and the dynamic conversion of these concepts at runtime. To analyze the impact of this mapping and avoid some of its consequences such as security holes and conflicts, the platform provides a set of tools to verify authorization attribute alignment, to validate local policies (for each organization), and to resolve authorization decision conflicts between them. Section 4.2.1 gives an overview on the administration mapping tool and Section 4.2.2 describes the dynamic conversion of authorization attributes.

4.2.1 Administration process and tools

In order to structure collaboration between organizations governed by different authorities, a consortium is needed. The consortium is set-up with a consortium agreement where all domains agree on:

- the resources and actions they accept to open to external users
- categories of users/roles having cross-domain needs
- aspects/attributes having a cross-domain usage

An authorization ontology shared between all the consortium members/organizations is necessary to unify authorizations.

Figure 3 illustrates the offline process and tools that are used to map authorization attributes.

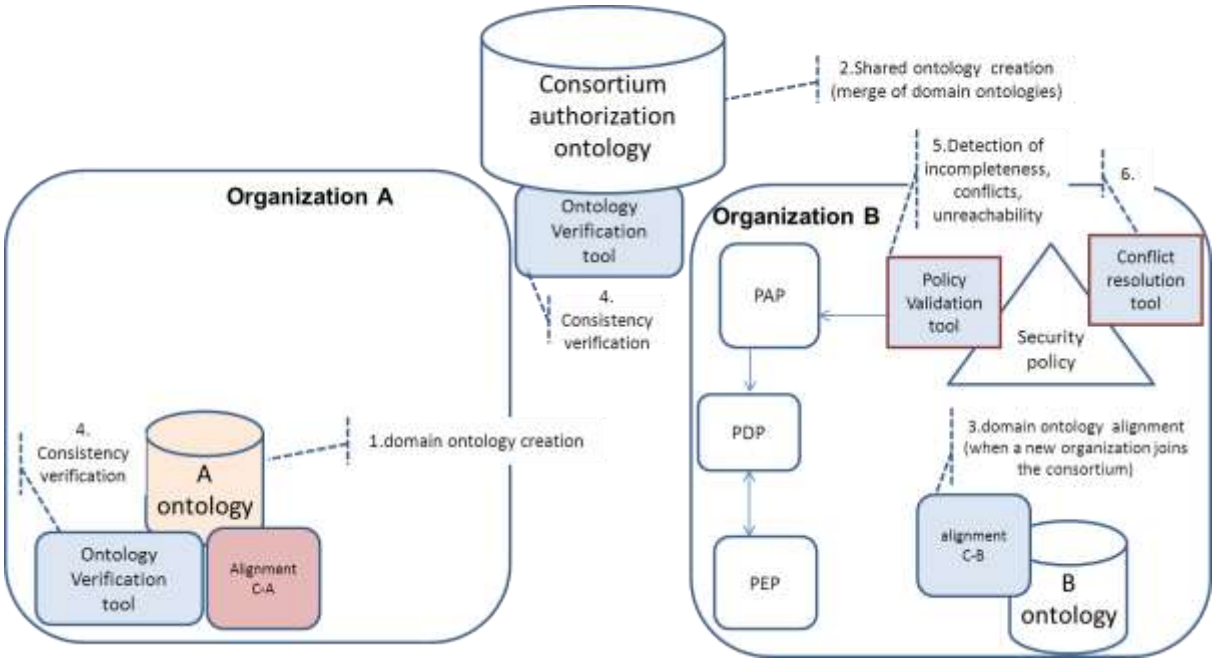

Figure 3 Authorization attributes alignement: process and tools

The ontology is defined after the consortium agreement with a bottom up approach following these steps:

1. Each organization administrator creates its domain ontology
2. The consortium administrator merges the domain ontologies into a shared ontology
3. If a new organization joins the consortium, its domain ontology needs to be aligned with the shared ontology

After merging or aligning ontologies, an ontology verification tool is used to check the consistency of the shared ontology and the alignments (step 4). Mechanisms related to AU2EU ontology creation, alignment, and verification are respectively descried in sections 4 and 5 in D3.1.1.

An automated validation tool is used by local policy administrators to analyze the behavior of policies (step 5). The validation tool leverages the features of ASP (Answer Set Programming) solvers in performing various logical reasoning and analysis of XACML policies, such as detecting security holes (policies incompleteness), and conflicts. The transformation of XACML3.0 logic to ASP and the approach of its usage in policy validation is described in D3.1.1, section 6.

A conflict resolution tool is used when conflicts are detected (step 6). The conflict resolution approach is described in D3.1.1, section 7.

Design and technical interfaces of each of the local policy validation tool and conflict resolution tool are respectively described in Section 6 and Section 7 of this document.

4.2.2 Runtime ontology conversion

The ontologies and their alignments are stored in distributed repositories (see Figure 3). Aligned attributes can be accessed by querying this repository at runtime. The core of this service is based on an ontology reasoner allowing to perform attribute conversion between organizations (see more details in D3.1.1 section 4). Examples of queries that may be used for attribute conversion:

- Is attribute *a* from organization A equivalent to attribute *b* from organization B?
- What are the attributes from organization A that are equivalent to attributes from organization B?
- What are the attributes (from all organizations) that are equivalent to the attribute *a* from organization A?

Figure 4 illustrates Online/Runtime ontology conversion process and tools.

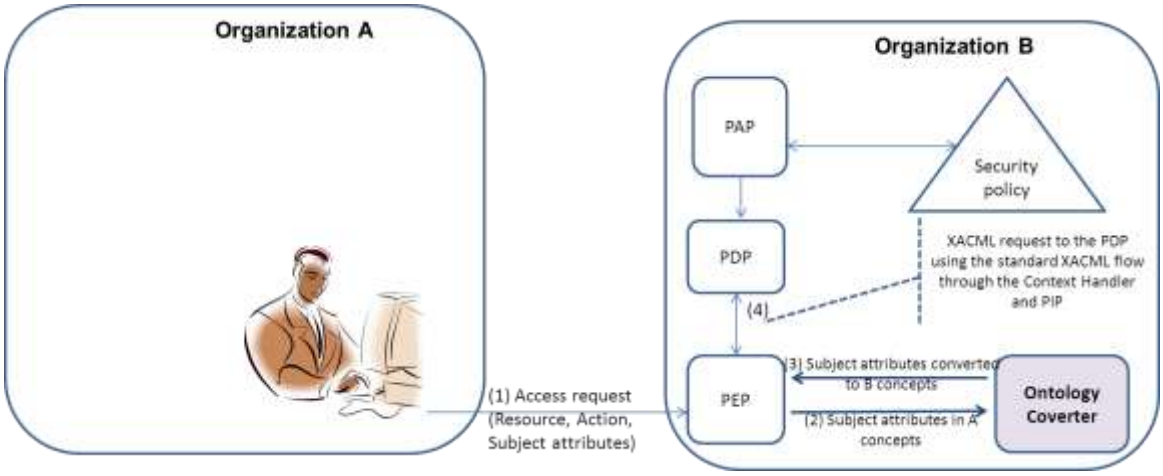
Organization A
Organization B
PAP
Security policy
PDP
XACML request to the PDP using the standard XACML flow through the Context Handler and PIP
(4)
(3) Subject attributes converted to B concepts
PEP
Ontology Coverter
(2) Subject attributes in A concepts
(1) Access request (Resource, Action, Subject attributes)

Figure 4 runtime conversion of authorization concepts

When a user in an organization A makes an access request to a resource and action of a service provider in an organization B, The PEP in organization B asks the ontology converter to convert the subject attributes expressed in organization A concepts (vocabulary) to organization B concepts. The PEP, then creates an XACML request to the PDP and requests the PDP to compute a decision, using the standard XACML flow.

4.3 Integrated architecture

One of the main objectives of AU2EU is to define and implement an architecture that integrates data-minimizing authentication protecting user privacy into a traditional authorization system such that the overall system is user privacy protecting.

In D3.2.1 we discussed the challenges and the possible solutions of such integration. Section 6 of D3.2.1 describes the proposed solution to integrate a XACML system with an ABC4Trust system. In this section, we base on this integrated architecture to describe a whole integrated flow that includes in addition a dynamic conversion component to take into account the multi-organizational context. It is important to recall that the architecture and mechanisms described in D3.2.1 need to be understood before reading this section.

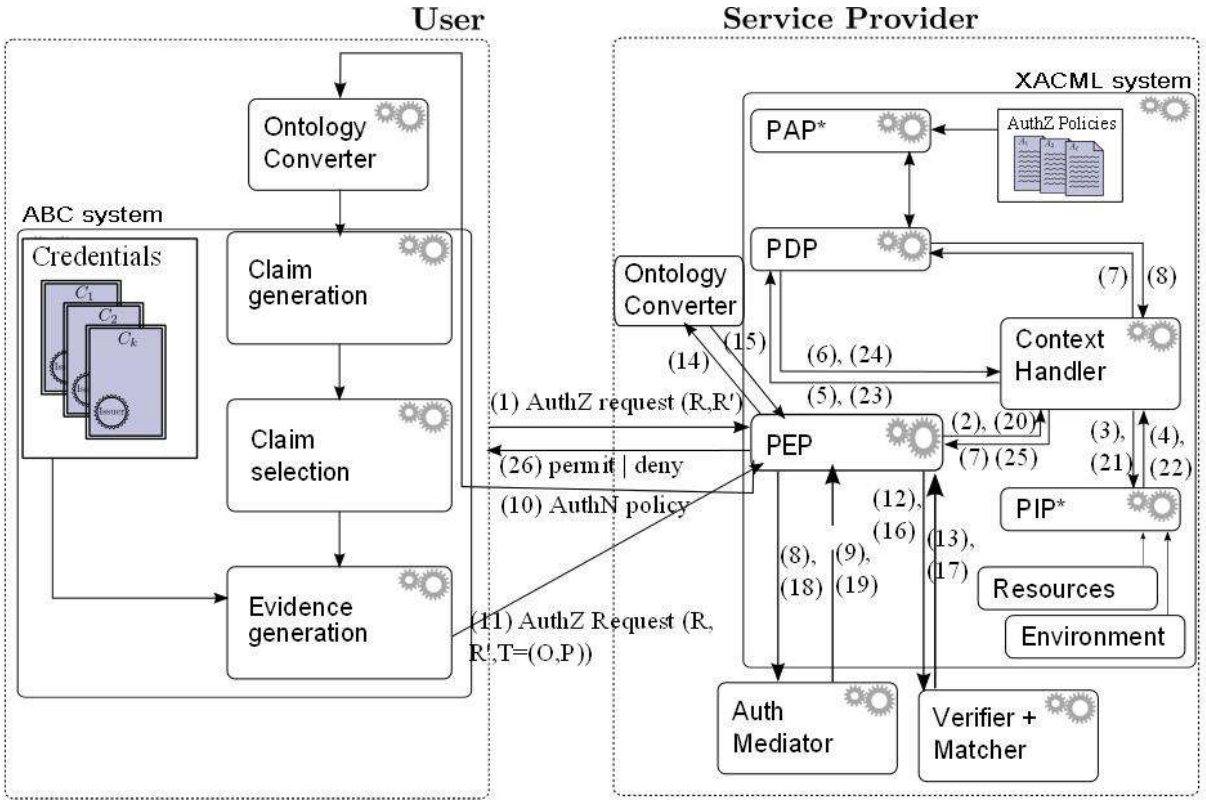

Figure 5 Integrated architecture

Figure 5 illustrates the architecture integrating each of the authentication framework, authorization framework, and dynamic ontology conversion. In summary the building blocks of the architecture are the following:

- An XACML framework on the service provider side (relying party). In order to capture the functionalities that are needed to integrate the authorization system with the ABC authentication system, the XACML system was extended. Main extensions consist in modifying the PDP and the standard flow to support mapping of authN and authZ policies and dynamic derivation of attributes representing authentication policies.

- An ontology converter on each of the service provider and the user side in order to convert authorization attribute concepts used in the user's organizations (identity providers) to authorization attribute concepts used in the service provider organization and vice versa.

- An ABC4Trust client system in the user side whose role is to dynamically generate tokens including assertions based on user's credentials and cryptographic proofs. Tokens are generated in a way that fulfills authorization policies and protects user privacy.

- An AuthMediator that is needed to map authorization policies (standard XACML policy) to authentication policies (ABC policy) and to generate a standard authorization request from authentication attributes. This component is integrated with the PEP on the service provider side.

- A Verifier, which is an authentication component on the service provider side that verifies the cryptographic proofs included in tokens. The verifier is able to extract the assertions of a to-

ken and send them to the PEP. This component is integrated with the PEP on the service provider side.

More conceptual details about the XACML extensions and the AuthMediator can be found in D3.2.1, section 6. Each of the verifier and ABC4Trust client are described in D2.1.1.

In the following, we describe the message flow related to an access request of the user for a resource provided by a service provider. The main idea of the solution is to enable the user to request the authorization system for an authentication policy before delivering his attributes to the authorization system, which avoids the user to reveal all his attributes. The user gets access to a resource in two main steps:

1. The user makes an access request, in form of a regular XACML AuthZ request that is processed by the XACML framework on the service provider. A 'deny' is returned, along with an AuthN policy which must be satisfied if access is to be granted. The AuthN policy need to be converted to the user's identity provider vocabulary in order to be employed by the user's ABC4Trust system for producing appropriate claim/evidence in the form of an ABC4Trust presentation token. The sequence diagram in Figure 6 details message exchange of this first part of the flow (called request without token). This approach requires the PDP to be modified in order to capture the functionality of extracting the reference attributes for the AuthN policies which need to be fulfilled in order to make an access decision. The mechanisms for extraction. Mechanisms for extracting authN policies from authZ policies are detailed in D3.2.1.

2. The claim/evidence is sent to the service provider, along with a repeated access request. The "Verifier" verifies the claim/evidence by determining whether associated the cryptographic elements support the exposed claims. The" verifier" verifies the parts of the policy against the claim and provides this information to the PDP through the PEP via a standard XACML request after using the ontology converter to convert the claims to the service provider vocabulary. A final access decision is made and communicated back to the user. The sequence diagram in Figure 7 details message exchange of the second part of the flow (called request with a token).

The detailed message flow is illustrated in Figure 5 as follows:

- The user makes an access request to a resource "R" and an action "R'" (step 1).
- The PEP creates an XACML request to the PDP to compute a decision using the standard XACML flow through the Context Handler and PIP. We assume that nothing is known about the requesting user U, that is, there are no subject attributes contained in the XACML request to the PDP. The PDP computes a decision for the access request, which will be 'deny', and derives the required user attributes (step 2-5).
- The PDP returns "deny" and a data structure comprising attribute requests, expressed as a disjunctive normal form. This data structure will be called A. Its semantics is that only one of the AuthN policies composed based on the minterms of this structure needs to be fulfilled by the user in order that the PDP be able to make an access decision (step 6-7).
- The PEP requests the AuthMediator component to translate the special attribute request A to an authentication policy A' expressed in the ABC4Trust format. The design and interface of

this component are described in Section 8. The obtained AuthN policy comprises a list of alternatives, corresponding to the minterms, and the user is free to choose which alternative they will provide a claim against to preserve their privacy (step 7-9).

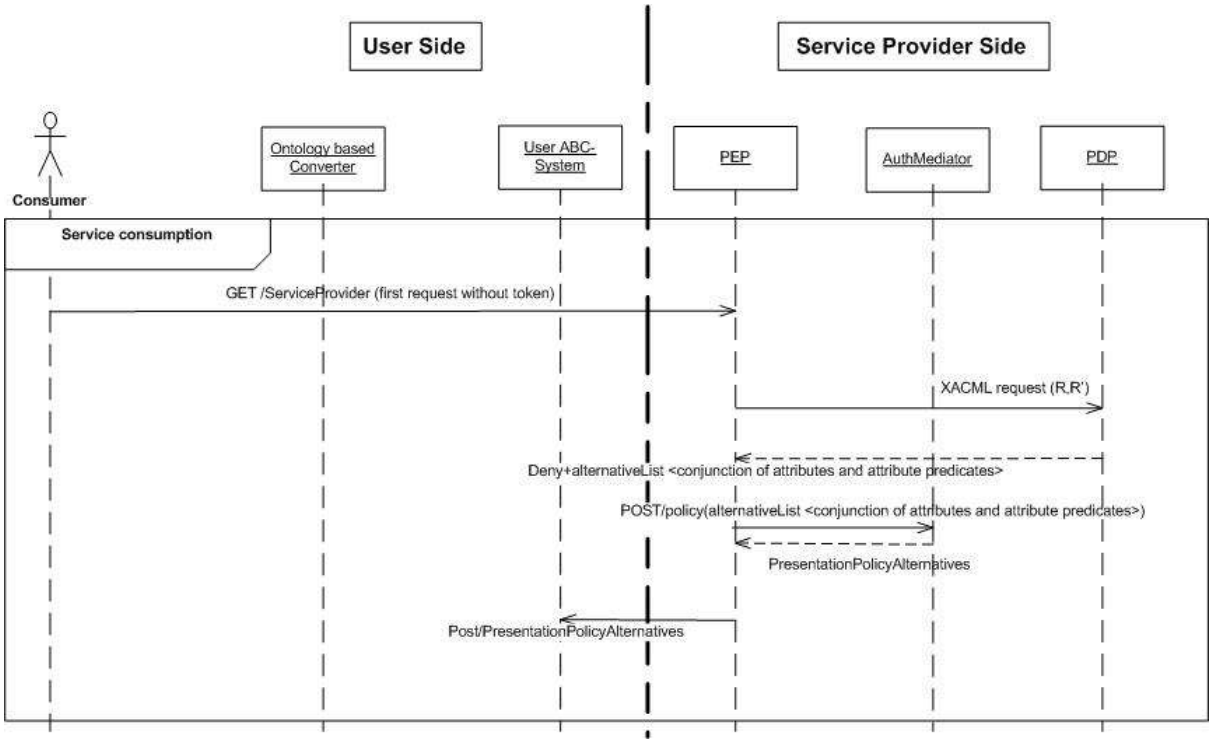

Figure 6 Request without a token

- The PEP sends A' to the user in order to communicate to the user which attribute assertions need to be released in order that an access decision can be made (step 10).

- The user's ontology converter performs an ontology-based conversion for transforming the presentation policy A' expressed in the vocabulary of the service provider to A'' expressed in the vocabulary of the user (the vocabulary used by the user's issuer). This conversion cannot be done at the service provider side for the reason that the service provider ignores the vocabularies of the user identity providers.

- The user's ABC4Trust system then computes claims candidates O_1, …, O_n, based on the user's credentials, that can fulfill A''. One of these claims is chosen, let it be called O. The ABC4Trust client creates a cryptographic evidence P for O, and O and P together form the presentation token T= (O, P).

- The user makes another access request to the resource R and action R' of SP and attaches the presentation token T= (O, P) to the request. (step 11).

- The PEP receives T and asks the verifier (authentication component at SP) to verify the cryptographic proof P of T. If the proof verifies, the verifier indicates to the PEP that the assertion is valid and sends the set of attribute values and predicates included in O to the PEP (step 12-13).

- The user's ontology converter performs an ontology-based conversion for transforming O to O' based on the mapping from the user's vocabularies to SP's vocabulary. The presentation token description is expressed, like O, in the ABC4Trust language (step 14-15).

- The PEP calls the Verifier for matching the presentation token description O' against a presentation policy A'. Let matching == success express that O' fulfills the policy A'. In this step, the ABC4Trust service identifies the attribute predicates that are false and attribute values and predicates that are undefined and send them to the PEP in addition to the attribute values and predicates included in O' (step 16-17).

- If matching == success, attribute values and predicates determined during the previous step are translated to a syntax close to XACML syntax for representing attributes (step 18-19).

- The PEP constructs an XACML request to(R, R') and the list of subject attribute values and predicates sent by the ABC4Trust translator in the previous step and requests from the PDP a decision. The PDP computes the decision using standard XACML policy evaluation. The decision by the PDP is enforced by the PEP (step 20-26).

4.4 Conclusion

This section described an overview of the components of the AU2EU eAuthorization framework and detailed their role and their integration. The following sections put a focus on the design and interfaces of these components: the ontology conversion service, the local logical-based XACML validation tool, the cross-organization conflict resolution tool, the AuthZ/AuthN mediation service, and the XACML framework. The XACML framework that is implemented and provided in this version of deliverable (D3.3.1) is a standard XACML 3.0. The XACML modifications briefly described in Section 4.3 and detailed in D3.2.1 will be implemented and integrated in the second version of the deliverable D3.3.2.

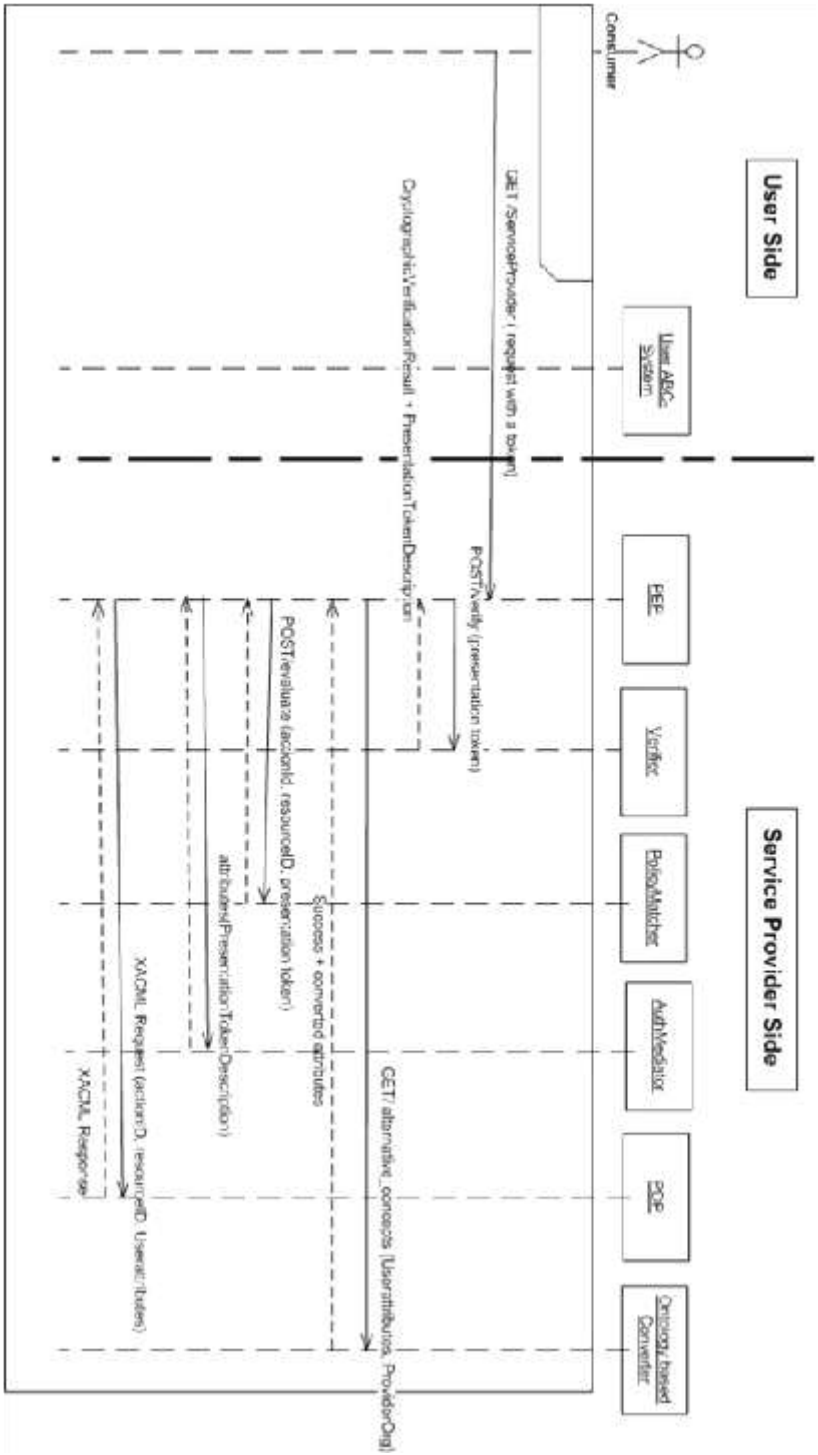

Figure 7 Request with a token

5 Ontology based conversion service

This section details the software architecture of the AU2EU ontology conversion tool. We will start with a short introduction on the topic and the purpose of the tool. We continue with listing its functional as well as non-functional requirements. Thereafter, we sketch its high level design, that is, we briefly describe used components and technologies and how they will interact. Next, we describe the interfaces which other components will need to address if they want to integrate the ontology based conversion service. The following section provides an installation guide that helps setting up the service. We conclude the section with some examples for using the service once it is deployed.

The main goal of AU2EU is to enable entrusted unions between organizations. In order to cooperate, they need to share resources and infrastructure, which requires mechanisms for access control to be in place. One of the main challenges in access control with respect to a multi-organizational context is the use of different vocabularies for authorization attributes. Different terminology can be a limiting, sometimes even blocking, factor when systems need to be integrated. Therefore, we develop an ontology-based conversion service to bridge the gap between different vocabularies within an entrusted union (see D1.4.1, Section 6.5 and D3.1.1, Section 4).

5.1 Ontology

In this subsection we introduce the ontology based conversion related terminology and we discuss the structure of the ontology and how it is built as well as trust implications of using the ontology.

5.1.1 Definition of terms

We use the terms below with the following interpretation:

Concept, term: the basic entity used in a (domain) model/language.

Ontology, vocabulary: a collection of concepts, their properties and relationships between them. (See also D3.1.1, Section 3.1)

As mentioned in D3.1.1, within the context of AU2EU, we refer to vocabulary and ontology synonymously. In the description below we mainly use the word ontology for an inter-organizational shared structure, such as the shared AU2EU ontology, while the words vocabulary and term are used for concepts used within an organization.

Conversion: the act of converting a term in one vocabulary to a related term in another vocabulary.

Alignment: a mapping between terms from different vocabularies/ontologies. (See also Section 5.1.2)

Merging: to create a shared ontology by considering multiple vocabularies. (See also Section 5.1.4.)

5.1.2 Ontology structure

From the introduction of this section we see that the key goal of the ontology-based conversion service is to enable a relying party (the service provider) to understand and be able to trust attributes provided by some issuer and vice versa for users to understand which of their credentials can provide the attributes required by the relying party.

The architecture of the authorization and authentication framework in general and that of the ontology-based conversion service in particular support different methods for structuring and establishing the ontology. For example, a relying party and attribute provider could directly align their terms re-

sulting in a peer to peer alignment between these two entities. This, however, requires alignments between (terms of) all cooperating entities. Alternatively several entities can agree on common concepts so each only needs to align their vocabulary with the shared concepts. Figure 8 shows the resulting alignments with respect to one concept used by six organizations each with their own terms.

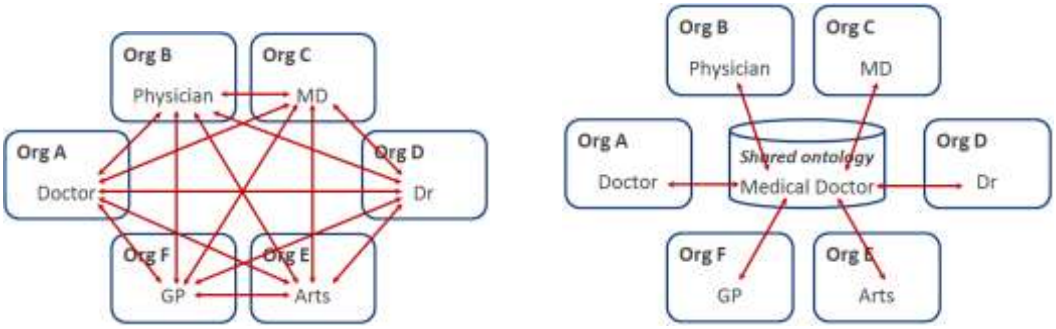

Figure 8 Peer to peer alignment of terms vs a shared ontology

As can be seen from Figure 8, using a peer to peer alignment of terms the size of the alignment grows exponentially in the number of participating entities versus a linear growth with a shared ontology. On the other hand, use of the shared ontology requires all entities to agree on a concepts meaning. As such a combination of these methods is needed to ensure conversion of terms that is efficient yet also practical to set up. The method we thus suggest (though any approach is supported) is that concepts that an entrusted union can agree on are placed in a shared ontology while peer to peer alignment (or a shared ontology of another, possibly smaller, entrusted union) is used for terms which have not (yet) been introduced as a concept in the shared ontology.

5.1.3 Trust in converted attributes

The use of the ontology-based conversion service has a potential impact on the trust relationships and on thus on the level of assurance, see [1] and deliverable D4.1.1, of the attributes. The entity running the ontology-based conversion service is added as a trusted entity. As a relying party can potentially run the ontology-based service themselves needing to trust this service is not a large limiting factor. Additionally the relying party has to trust that the alignment of the terms is done correctly.

The implications of trusting the alignment depends on the type of alignment; peer to peer or via the shared ontology. With an attribute mapped via a peer to peer alignment, the attribute provider and the relying party have together directly determined that the term used by the provider is appropriate for the term required by the relying party. If mapped via the shared ontology the relying party trusts that (besides aligning correctly itself) the issuer of an attribute, in addition to assigning the attribute to the right entities, has also correctly aligned this attribute to the shared ontology. As this issuer is apparently an authority on this attribute this is not an unreasonable assumption.

Here we assume that the alignment is correctly created through a fully trusted method. If in a giving setting this procedure is not always fully trusted it is advised that the requirements for levels of assurance, in addition to the usual requirements on how attributes are issued and how possession of attributes is proven, are extended with requirements on how the alignment of the attribute is performed.

We conclude that while participants, and especially the relying party, should be aware of the implications of using the ontology-based conversion, this use does not limit the level of trust and level of assurance that can be achieved.

5.1.4 Building the ontology

Each entrusted union will need to build its own shared ontology. The "richer" this ontology is, i.e. the more terms used by the collaborating entities it captures in concepts and the more precisely it does this, the less additional peer to peer alignments will be needed thus the more efficient the conversion will be. Entities can define shared concepts by "merging" their vocabularies as mentioned in deliverable D3.1.1 "Tools for semantic mapping of attributes". Note that this requires examining the definition of the terms and their relevance to the collaboration rather than only considering the names of the terms. We illustrate this with an example. Consider *Care center A* uses term *Doctor*, *Clinic B* uses term *Physician*, *Hospital C* uses term *Surgeon* and *University U* uses term *Doctor*. While *A.Doctor* and *B.Physician* are quite similar, a *Surgeon* at *Hospital C* has relevant distinguishing features and additional rights (e.g. use of the operating room) that not all MDs have. *U* uses the same term as *A* yet the meaning of the term is different; *U.Doctor* is a scientist while *A.Doctor* is an MD. The collaboration thus includes the concepts *MD*, *Surgeon*, and *Scientist* in its shared ontology, and *Surgeon* is made a sub concept of *MD*. *A* equates its term *Doctor* with *MD* as does *B* for its term *Physician* and *C* equates its term *Surgeon* with the shared concept *Surgeon*. Finally *U* states that its term *Doctor* is a sub concept of *Scientist*.

As mentioned the architecture is agnostic to method of creating the ontology. We suggest a hierarchical approach to maximize reuse and thus minimize the effort in setting up a (rich) shared ontology. At the top level there will be a very general ontology that contains terms the will be applicable in nearly all application domains. In our implementation we use schema.org as this most general ontology. Such a general ontology will not provide all terms needed within each application area. Thus application area specific extensions of the general ontology are made. To illustrate this we build an `entrusted union' AU2EU ontology with concepts relevant to several AU2EU use cases. Finally specific applications in the area may require yet more specialized concepts which can be captured in a further extension of the ontology. For example, we will consider applications in the pilot application areas to see if such additions are needed to the AU2EU ontology. The hierarchical approach of building the ontology allows new entrusted unions to start from an available ontology that is as close as possible to their applications and/or application area, minimizing the effort needed to build a rich shared ontology.

Automated support for generation of alignments may be desirable, especially when creating peer to peer alignments. This may help reduce the required effort and keep the process manageable. Still, the automation should not unduly decrease the trustworthiness of the resulting conversion. Possible approaches include alignment sharing based on reputation [2], [3].

5.2 Requirements

After analyzing the use case requirements, we have identified three functional requirements and non-functional requirements. In detail, the functional requirements are:

- The conversion service MUST be able to *identify* similar concepts to a given concept. The similarity can either be equality, or more generic concepts.
- The conversion service MUST be able to *verify* a similarity relationship between two given concepts.
- The conversion service MUST provide interfaces to efficiently handle identification and verification, i.e. means to bundle requests.

- The conversion service MUST allow each organization participating in the entrusted union to maintain their own vocabulary and mappings.

The non-functional requirements are:

- The conversion service SHOULD be interoperable with other software components
    - It SHOULD be based on open standards.
    - It SHOULD have a modular design.
    - Its components SHOULD have clear interfaces, with clearly separated concerns.
    - Its components SHOULD be replaceable with other components adhering to the interface contracts.
- The conversion service SHOULD be able to run on commodity hardware.

5.3 Design

In this section we sketch the architectural design of the ontology conversion service. We start with briefly introducing the technologies that will be used. We continue with describing the composition of the components and the message flow between them. Finally, we detail the SPARQL queries that will be used for the conversion of the authorization attributes.

The technologies and frameworks we build our ontology conversion service on are:

Sinatra[1] is a, small, flexible, Rack-based library written in Ruby that can be used to create web applications and web services with minimal effort. Sinatra is open source, developed by a vibrant community and backed and used by renowned companies such as LinkedIn, Heroku, GitHub, Apple and BBC. The use of Sinatra is straightforward, and Web Services developed with Sinatra convince with their simplicity. We will use Sinatra to develop our conversion tool REST API.

SPARQL [4] is a W3C Recommendation to query RDF data. In addition to its query capabilities it also allows for querying federated data sources, supports multiple output formats (e.g., JSON, XML, CSV) and allows for modifying triple data using SPARQL statements. The standard's current version is 1.1. For the ontology-based conversion tool we utilize its SELECT and ASK query-form. The SELECT query form answers queries for similarity concepts and the ASK query form can be used to verify specified query patterns.

Apache Jena[2] is an open source Java framework for building Semantic Web applications. Amongst others, it provides an RDF API, an RDF triple store, a SPARQL query engine and endpoint as well as an inference engine. The Apache Jena components that we will use for the conversion service are: JenaTDB, which is Jena's native triple store implementation; ARQ, which is the SPARQL query engine in Jena and Fuseki, which allows for accessing the ARQ and JenaTDB in a RESTful way over HTTP. It is worth noting that ARQ provides the means for federated queries. Apart from that we will

[1] http://www.sinatrarb.com/intro.html
[2] http://jena.apache.org/index.html

use the Jena's inference API combined with an OWL Micro Reasoner. This will allow us transitive inference on concept equality as well as subClassOf / subPropertyOf relationships. Apache Jena provides more expressive reasoners as well, however, they are less performant.

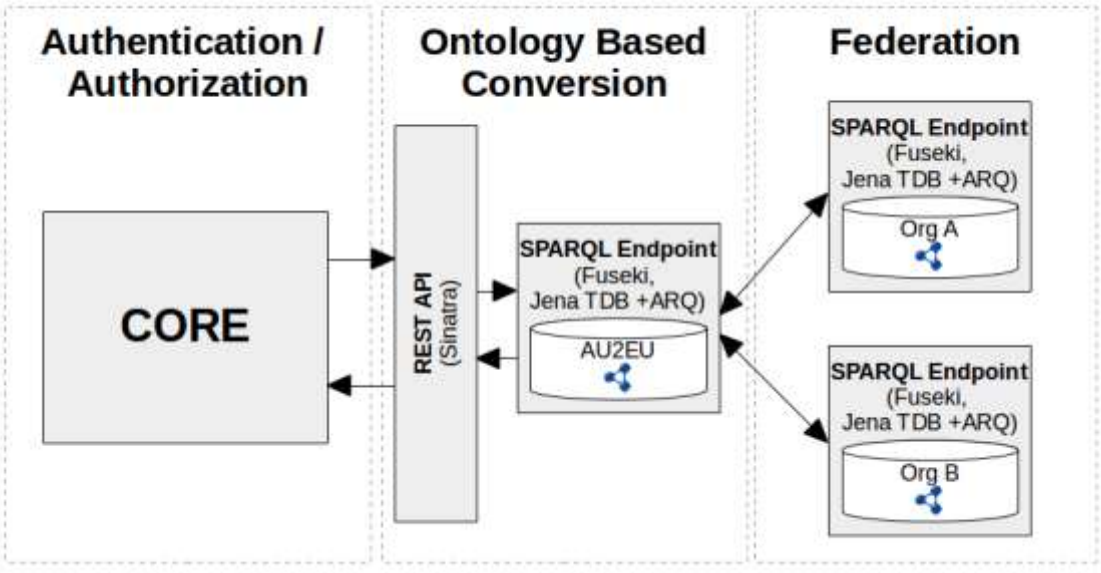

Figure 9 Software Architecture of the Ontology Based Conversion Service

As introduced in D1.4.1 and D3.3.1, the ontology conversion service will be used at two different times during the authorization / authentication workflow. The *User* searches alternative concepts to a given concept, i.e. to a credential-certified attribute they possess. The *Relying Party* verifies the similarity between two given concepts to ensure that a proof given for one concept also satisfies the requirements of the presentation policy. Figure 9 schematically illustrates this communication between the authentication / authorization core and the conversion tool. It also depicts the two main components of the conversion service, namely a Sinatra-based REST API and a Jena-based SPARQL endpoint.

The main purpose of the REST API is to add an abstraction layer between the SPARQL endpoint and the application core. It receives the parameters from the core, transforms them to a SPARQL query, executes the query and returns the result back to the core. This additional abstraction layer renders it easy to exchange the mapping technology, for example one could switch to a primitive hash map or a simple database.

The SPARQL endpoint stores the vocabularies and provides the means for querying and reasoning about them. We chose to integrate the SPARQL via the Fuseki REST HTTP interface instead of using the native Java libraries, because the REST HTTP interface is technology agnostic. This potentially allows for exchanging the technology used to query the SPARQL endpoint. The drawback of this solution is a performance loss due to the added network overhead. Apart from that, Fuseki takes care of data federation, i.e. querying over different repositories. This enables each organization to use their own vocabulary, hosted on their own servers in their own repository, without limiting the functionality of the conversion service. The alignment between the shared domain ontology and an organization's ontology can also be stored in their own repository. Please note that in AU2EU we will only use

one SPARQL endpoint, which will host both, the shared domain ontologies as well as the alignments. This approach reduces the implementation complexity, while it still demonstrates the idea of the conversion very well. Adding more SPARQL endpoints is trivial.

Our REST API will communicate data in JSON. JavaScript Object Serialized (JSON) data is less verbose than XML, consumes less bandwidth and parsers are more performant. Apart from that, the ontology conversion service will not send structured data. Alternatively, protocol buffers offer even higher performance, however, they are not widely adopted yet.

As previously mentioned, two different types of SPARQL queries form the core of our conversion service, a SELECT query and an ASK query. When the conversion service receives a request, it will parse the parameters and transform them to an according request. The SELECT query will be of the following form "*SELECT ?similar_concept a <requested_concept_uri>*". The keyword "*a*" is a SPARQL shortcut for "*http://www.w3.org/1999/02/22-rdf-syntax-ns#type*". Thus, the above query will find all concepts which are equal to the requested concept. It will also find inferred types due to the used OWL Micro Reasoner. Additionally, one may declare which concepts they own themselves and the vocabulary of which organization the alternative concepts should origin. These two parameters are optional.

In contrast, the ASK query will be of the form "*ASK <concept_a_uri> a <concept_b_uri>*". It returns a boolean value which indicates whether this pattern can be found in the current ontology or not, in other words whether these concepts are related or not. Again, since we are using a reasoner, transitive relationships between concept/property types will be manifested and will provide a correct result.

5.4 REST API

In this section we describe the REST API for the ontology-based conversion service. In Section 5.6 we provide examples for both calls.

Method Name	*GET /alternative_concepts*	
Method Definition/Description	This HTTP GET method retrieves all similar concepts for a given URI (equal or more generic). A concept is identified by a URI.	
Method input attributes		
Name	***Type***	***Description***
required_concepts	A set of comma-separated, URL-encoded URIs	The URIs denoting the concepts to find alternative concepts for.
source_organization	Optional. A URL-encoded URI string.	If this URI is specified, it will only return the concepts from the vocabulary of the specified organization.
Method output attributes		
Name	***Type***	***Description***
	JSON Object	A JSON Object that contains the response of the call. It consists of the following attributes: • ***"success"***: A boolean flag indicating whether the request was executed successfully.

		• An attribute for each requested URI, containing an array of alternative Concept URIs, i.e. "required_uri" : ["alternative_uri1", "alternative_uri2", …]

Method Name	*GET /alignments/verify*	
Method Definition/Description	This HTTP GET method validates a given alignment between two concepts.	
Method input attributes		
Name	***Type***	***Description***
alignment	A named map where the keys are URL-encoded concept URIs and the values are the matching encoded concept URIs.	The alignment to verify. It may consist of more than one matching concept URI pair, i.e. *encoded_concept_uri=matching_concept_uri*.
Method output attributes		
Name	***Type***	***Description***
	JSON Object	A JSON Object that contains the response of the call. It consists of the following attributes: • ***"success"***: A boolean flag indicating whether the request was executed successfully. • ***"valid"***: A boolean flag indicating whether all alignments are valid.

5.5 Installation guide

This section will provide a schematic installation guide to install the ontology-based conversion service. We intended to develop this tool as technology agnostic as possible, therefore there are many different ways to achieve an installation, for example it does not really matter whether the operation system of the server is Windows or Unix based, nor does it matter which JVM you will install.

5.5.1 Hardware requirements

The ontology conversion service is designed to run on a commodity server with very little hardware requirements. We recommend a server with at least:

- 2 GB of free RAM available.
- 3 GB of free hard disk available (i.e. enough to install Ruby, Sinatra, Fuseki, Jena, Java, and load the ontology)

5.5.2 Software requirements

In order to run the conversion service, you will need to install the following software on your system:

A Java Virtual Machine (JVM), version 6 or later. Depending on your machine's architecture, you will need to choose between a 32-bit or 64-bit JVM. We recommend to use a 64-bit implementation. Installation instructions for different Java Versions can be found here:

- Oracle Java: http://www.oracle.com/technetwork/java/javase/downloads/index.html
- OpenJDK: http://openjdk.java.net/install/

A Ruby interpreter, version 1.9.2. or later.

- For Windows based OSes, the recommended way to install it is to use the WindowsRuby installer: http://rubyinstaller.org/
- For Unix based OSes, we recommend install Ruby using the Ruby Version Manager script, which can be found here: the http://rvm.io/rvm/install

An Apache Jena installation as well as a Fuseki Server. Download and installation instructions can be found at the project homepage:

- https://jena.apache.org/download/index.cgi

5.5.3 Installation process

The installation of the ontology-based conversion service consists of the following steps:

- Start Fuseki and load the ontology. Instructions on starting Fuseki can be found here (http://jena.apache.org/documentation/serving_data/#getting-started-with-fuseki). Instructions on how to deploy the ontology can be found at the README within the conversion service archive.
- Download and extract the conversion service archive and extract it.
- Start the conversion service. To do so, execute "ruby conversion_service.rb" in the directory where the service files were extracted to.
- Additional configuration. The provided files use all default configurations and will work out of the box. The README file in the conversion service archive will explain modifications to the default configurations.

5.6 Examples

In this section we provide two examples of how to use the ontology based conversion service. We will use the UNIX-based command line tool *curl*[3] for our demonstration. Other clients that are capable of sending REST requests may be used as well. First, we will show how similar concepts using the conversion service can be found. In the second example we will validate the equality of two given concepts.

[3] http://curl.haxx.se/

5.6.1 Example usage – identifying similar concepts

This example demonstrates how to get alternative concepts for the property *http://schema.org/email* of the organization *http://au2eu.eu/aal* .

The following request to the ontology conversion service:

```
curl -X GET -G \
http://localhost :4567/alternative_concepts? \
required_concepts=http%3A%2F%2Fschema.org%2Femail &\
source_organization = http%3A%2F%2Fau2eu.eu%2Faal%2F
```

Will produce this JSON response:

```
{
  "success":true,
  "http://schema.org/email":[  http://au2eu.eu/aal/mail ]
 }
```

5.6.2 Example usage – validating similarity of concepts

This example verifies whether there two concepts are related.

The following request to the ontology conversion service:

```
curl -X GET \
http://localhost :4567/ alignments/verify? \
http%3A%2F%2Fschema.org%2Femail=http%3A%2F%2Fau2eu.eu%2Faal %2Fmail
```

Will produce this JSON response:

```
{
  "success":true,
  "valid":false
}
```

6 Local policy validation tool

6.1 Requirements

Local administrators of organizations specify XACML policies either manually or using automated XACML policy specification tools and need to ensure that the big number rules, policies and combination algorithms written in XACML3.0 correspond to what they intend to formulate in the overall policies. For example domain administrators need to ensure that no specific types of requests get an access decision that is contradictory with his intention, or that a "not applicable" answer is received for certain requests. In addition, they need to analyze the behavior of the specified policies after merging or mapping domain vocabularies, detect policies incompleteness, conflicting policies, unreachable policies, etc. More details about these anomalies are given in D3.1.1, section 6.1.1.

6.2 Design

In order to enable detection of the anomalies listed above, the idea is to build a tool that enables administrators of organizations to make "inverted XACML requests". Such tool is able to determine a set of possible requests that satisfy a set of XACML access control policies and a set of constraints/facts given as input for an intended access control decision (also given as input) (see Figure 10).

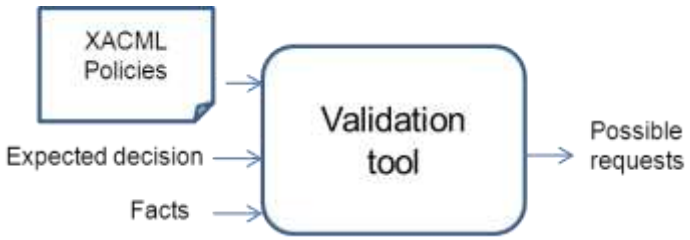

Figure 10 Validation tool

In addition to the XACML policy set that is needed to be evaluated, the administrator specifies the decision he expects from this policy set; permit, deny or not applicable. He may also specify a full or partial evaluation of elements from the XACML Policy set (that we call facts) that the validation tool must consider to perform the inverted policy request. Examples of partial evaluation of policy set elements could be:

- The attribute "subject-citizenship" can be evaluated to "FR" or "AU"
- The attribute "subject-citizenship" can only have 0 or 1 evaluation at a time.

Inverted requests allow administrators to answer the following questions: *Is there a request or a set of requests, given an optional set of constraints on given attributes (attributes of the main XACML categories: user, action, resource , environment), that result in the following decision: Permit, Deny or Not applicable ?*

In order to perform validation of an access control policy written in XACML V3, An ASP solver is used. XACML descriptions are translated into an ASP program written in an ASP language, in order to be solved by the ASP solver. Each of the access decision and facts given as input to the solver are expressed as ASP constraints (see Figure 11).

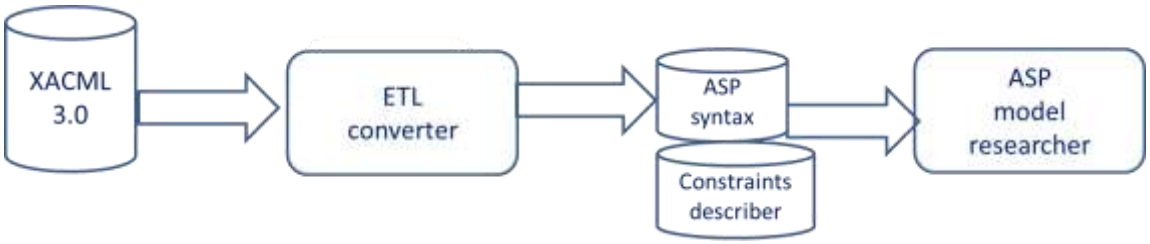

Figure 11 Validation tool design

The validation process consists in a transformation of an XACML 3.0 policy set into an ASP program, converting all XACML supported elements into ASP rules. This ASP program is then completed with a constraint file, containing ASP constraints formatted in the same ASP language. These constraints include the facts and the expected result (example: the policy set must be evaluated to "Permit").

The resulting program is then sent to an ASP solver, which returns one or several models, allowing determining whether the given constraints could be satisfied over the policy. Requests that satisfy the expected result are determined from the models satisfying the constraints.

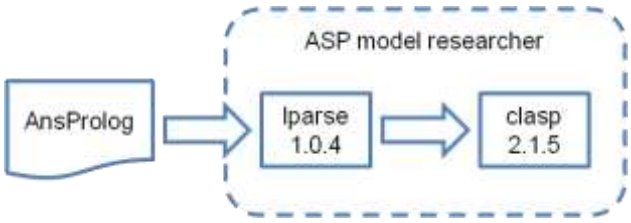

Figure 12 ASP model researcher detailed design

The ASP model researcher (see Figure 12) is composed of two ASP tools that are lparse and clasp.

lparse[4] is a front-end grounding tool, originally created for the answer set solver SMODELS, adding a layer of syntaxic sugar on the top of it. The language that lparse accepts is now commonly called "Answer Set Programming in Logic" (AnsProlog). It transforms an AnsProlog user program into an SMODELS notation. The version used is 1.0.4, under GPL licence.

clasp[5] is an answer set solver, part of the Potassco project (Postdam Answer Set Solving Collection bundles tools for Answer Set Programming), for standard logic programs, using SMODELS notation. The version used is 2.1.5, under GPL licence.

6.3 Interfaces

Method Name	*Solve*
Method Definition/Description	Solves the satisfiability of the input constraints, according to the given facts and a given XACML Policy set, and returns at the most the given number of possible compliant requests.

[4] http://www.tcs.hut.fi/Software/smodels/
[5] http://www.cs.uni-potsdam.de/clasp/

Method input attributes		
Name	***Type***	***Description***
policySet	File	A XACML PolicySet file
Constraints	File	An ASP file describing the constraints that must be considered for the validation
maxSolutions	Int	The max number of solutions that must be computed (0 for as many solutions as possible).
Method output attributes		
Name	***Type***	***Description***
Result	Text	A list of possible solutions, expressed under ASP facts format, each fact representing a partial evaluation of the policy set (at attribute or higher level).
Dependency with other components		

6.4 Installation guide

6.4.1 Hardware requirements

1 GB RAM, 50 MB Disk space

6.4.2 Software requirements

Java 1.7

6.4.3 Installation process

Unzip XACML_Analyzer.zip in a folder.

6.5 Examples

In this section we describe step by step a validation process.

The example below involves an access control policy in which users, having given roles, and being in a specific location, are trying to access machines, being themselves in some location (The XACML policy related to this example is detailed in Appendix A).

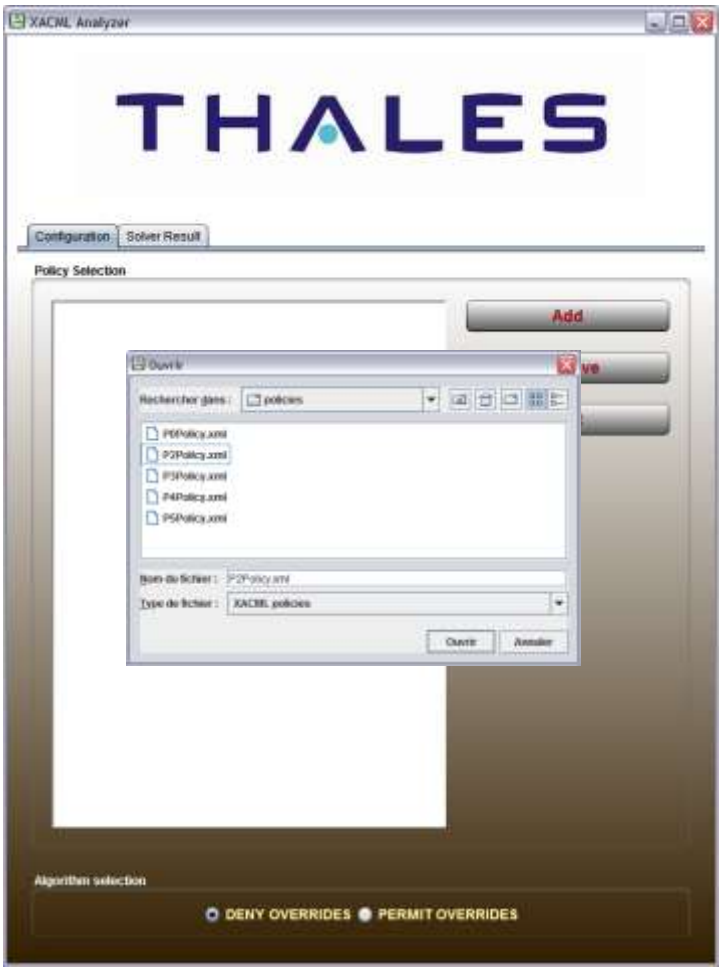

Figure 13 **XACML policies loading**

Figure **13** shows how the XACML policies set are loaded into the validation tool. These policies include:

- A policy that allows subjects, according to their roles, to perform some actions on any resource, within time ranges (see Policy1).
- A policy denying any access to some blacklisted machines (see Policy2).
- A policy representing inactivity of specific users during some date ranges, denying all of their requests (see Policy3).
- A policy granting exceptional access, for some users to some resources, within a given date range (see Policy4).

Figure 14 **Constraints file loading**

Figure **14** shows the loading of a constraint file, formatted in ASP.
Here is an example of constraint file, in AnsProlog:

is_role ("manager"; "operator"; "engineer"; "technician").
is_location ("Bilbao"; "Madrid"; "Barcelona"; "Valencia").
is_action ("operator"; "admin").
is_machine ("maq1"; "maq2"; "maq3"; "maq4"; "maq5"; "maq6"; "maq7"; "maq8"; "maq9").
is_user ("Maria"; "Lucia"; "Daniel"; "Hugo"; "Jordi").

0 {access_subject ("urn:oasis:names:tc:xacml:2.0:subject:role", Role): is_role (Role)} 1.
0 {access_subject ("urn:oasis:names:tc:xacml:1.0:subject:subject-id", User): is_user (User)} 1.
1 {access_subject ("urn:oasis:names:tc:xacml:1.0:subject:user-location", Location): is_location (Location)} 1.
1 {resource ("urn:oasis:names:tc:xacml:1.0:resource:resource-location", Location): is_location (Location)} 1.
1 {resource ("urn:oasis:names:tc:xacml:1.0:resource:resource-id", Machine): is_machine (Machine)} 1.
1 {action ("urn:oasis:names:tc:xacml:1.0:action:action-id", Action) : is_action (Action)} 1.
1 {environment ("urn:oasis:names:tc:xacml:1.0:environment:current-time", I) : is_integer (I)} 1.
1 {environment ("urn:oasis:names:tc:xacml:1.0:environment:current-date", I) : is_integer (I)} 1.

:- not val(overallPolicySet, permit).

Let's notice the following constraints:

- The different constant values that must be taken as possible evaluation for role, location, action, machine and user id.
- The smallest and biggest number of evaluations each attribute can have.
- The evaluation of the overall PolicySet must be "PERMIT"

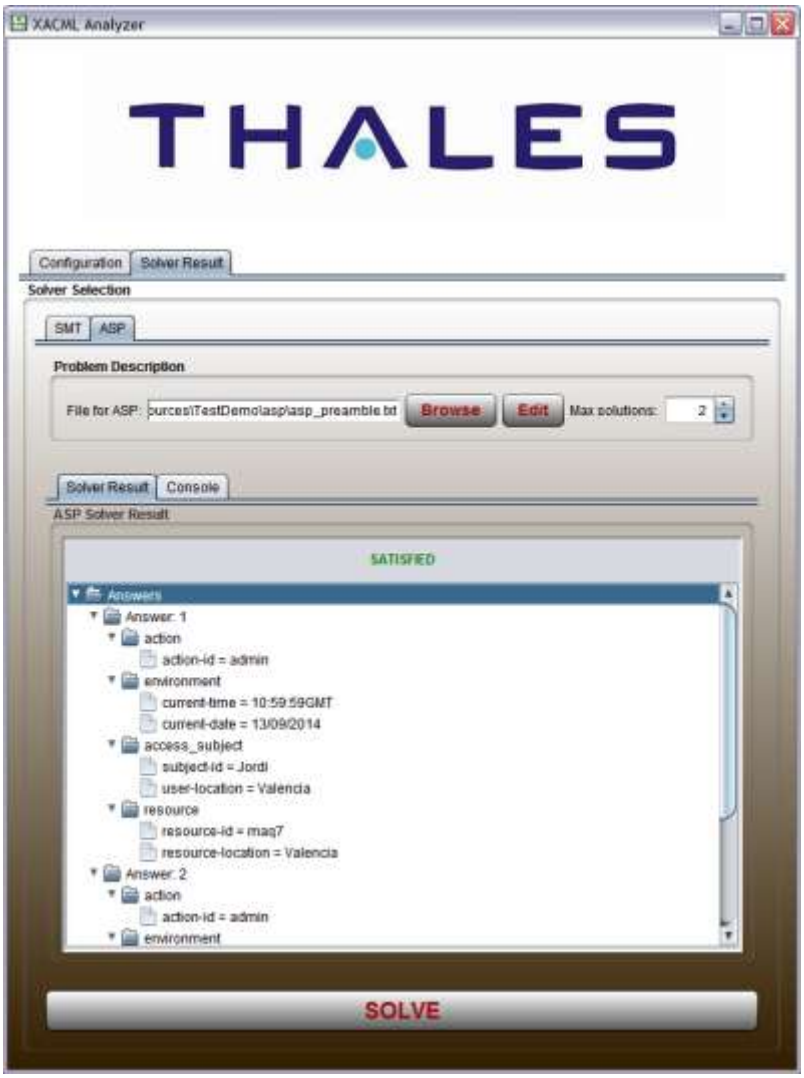

Figure 15 Formatted resolution

Figure 15 shows the resolution of the constraints over the policies, asking for 2 solutions at the most. These solutions are XACML request examples, taken from a combination of the different attributes mentioned in the policy and the different specified constant values, according to the constraints file. In the case where we are looking for request examples, solutions are formatted as a set of key/value association, grouped by attributes categories. In this execution, we can see that action "admin" for subject "Jordi" in "Valencia" will be allowed on the resource "maq7" located in "Valencia", the 13/09/2014 at 10:59:59GMT.

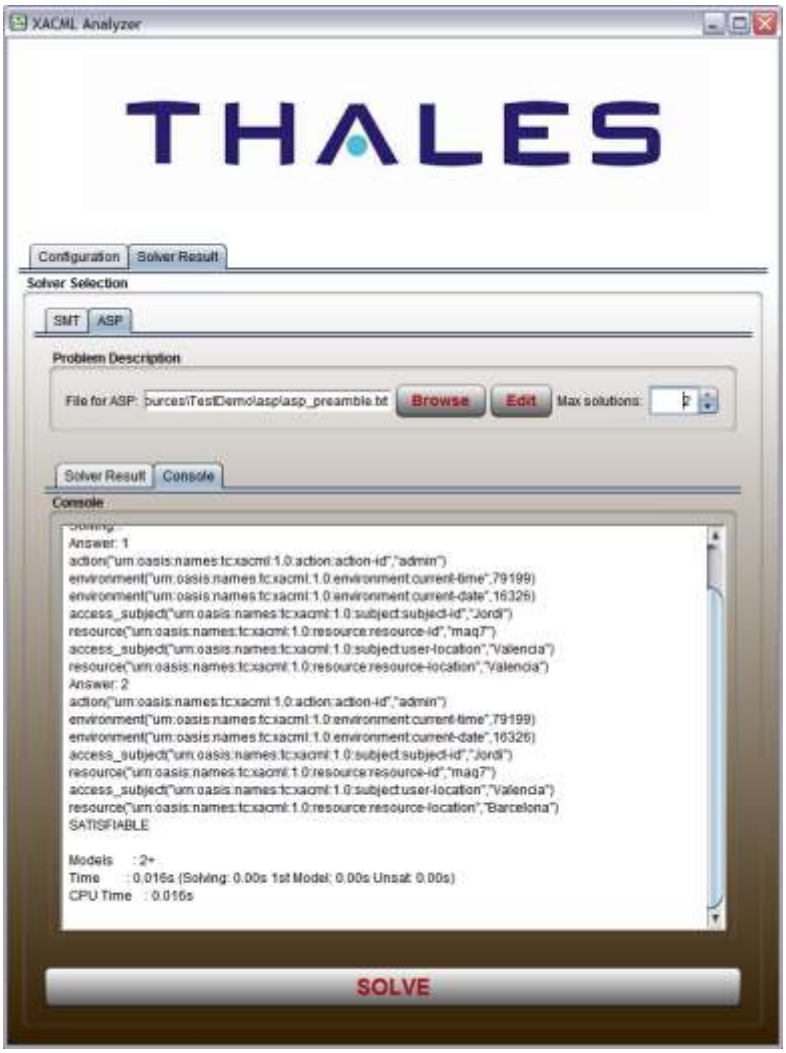

Figure 16 Raw solver output

Figure 16 shows the raw text output of the underlying solver, useful for the cases where the solutions would not be directly translated into key/value pairs, but would give other kind of information (e.g. the possible evaluation of a specific rule or predicate, satifiability of very specific constraints, etc.)

7 Policy conflict resolution tool

In this section we detail the components needed for our dynamic policy conflict resolution tool which has been introduced in D3.1.1, Section 7. We detail the high-level design and its internal message flow. Additionally, we provide generic installation instructions.

7.1 Design

This section gives a high-level overview of the implementation of the AU2EU conflict resolution tool and introduces the technologies that are used to deploy it.

Figure 17 depicts our integration approach of the AU2EU conflict resolution tool in the XACML data flow [5]. It omits components of the XACML data flow that are irrelevant for the conflict resolution. It also indicates which parts will be provided by SAFAX. We will add a conflict resolution component as a service. The data flow with conflict resolution enabled is as follows. The policy owners specify an access control policy for a shared resource. Note that several different policy owners may specify policies for the same resource. The owner includes an attribute in the policy's target which states that they are the owner of this policy. Additionally, they specify what type of conflict they are interested in using attribute/value pairs within the *PolicyIssuer* element (see D3.1.1, Section 7). This completes the policy definition phase. Policies are combined in a policy set with the combining algorithm agreed on for the collaboration and deployed. The next phase considers the use of the shared resource. A user issues an access request for the shared resource to the *Conflict Resolution Service*. The request passes the *PEP*, is forwarded to the *Context Handler* and after that forwarded to the *PDP*. The *PDP* obtains the policies relevant for this request from the *PAP*, evaluates these policies to an access decision and returns this decision to the *PEP* via the *Context Handler* and then back to *Conflict Resolution Service*. The *Conflict Resolution Service* returns the PDP's access decision to the user immediately and asynchronously starts the conflict detection.

In order to detect a conflict, the *Conflict Detection* component evaluates the request from the policy owner's viewpoint and compares the resulting access decision with the *PDP's* final access decision (see D3.1.1, Section 7.1). If the decisions mismatch in a way that the policy owner is interested in, a justification will be generated using the request, the conflicting policies as well as the policy combining algorithm. The last step is to notify the policy owner about this incident.

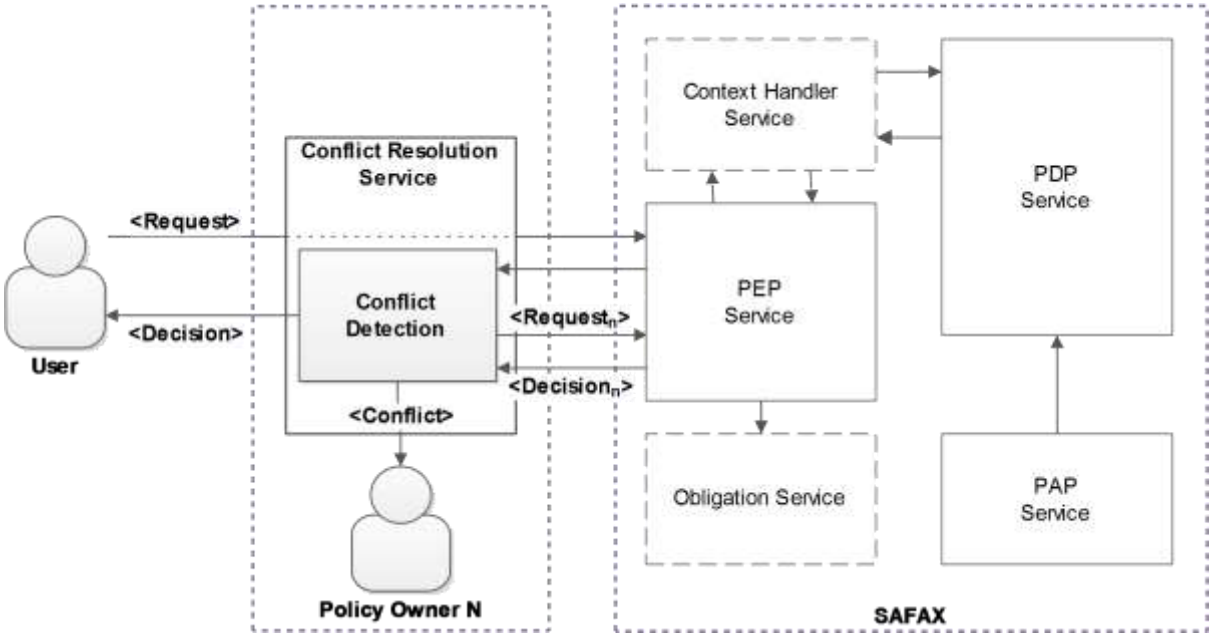

Figure 17 Conflict Resolution Tool - high level architecture

The technologies that are used to deploy the AU2EU conflict resolution tool are:

Apache Tomcat[6], Jersey[7] and MySQL[8] server. Apache Tomcat is an open source Java Servlet engine, which we use as foundation of our architecture. Jersey is an open source framework for developing RESTful Web services in Java. We use the Jersey library on top of Apache Tomcat. The MySQL community server is used for persisting state information.

Furthermore, we use SAFAX. SAFAX is an XACML based, architectural framework which implements XACML components as RESTful webservices. More information on SAFAX can be found in D4.2.1 or on the SAFAX webpage[9]. The conflict resolution version uses the WSO2 balana PDP library for evaluating policies. WSO2 balana is open source, based on Sun's XACML implementation and supports both XACML 2 and XAXML 3.

Figure 17 depicts the high-level architecture of the AU2EU conflict resolution(CR) tool. It is implemented as RESTful service. The tool consumes the XACML service components that are provided by SAFAX, such as the *PEP* or *PDP*. The CR tool is an extension to the existing architecture, thus does not influence the regular XACML workflow.

7.2 Interfaces

In this section we describe our conflict resolution tool, which will be a component in the SAFAX architecture. It will follow the conventions of the SAFAX architecture, namely that "all services are purely stateless and all information to the state is contained either with the request or stored in a per-

[6] http://tomcat.apache.org/
[7] https://jersey.java.net/
[8] http://www.mysql.com/products/community/
[9] http://security1.win.tue.nl/safax/

sistent database" and "Services communicated with each other either through JSON or XML messages".

Method Name	*POST /evaluate*	
Method Definition/Description	Evaluates an access request. Detects and conflicts and notifies interested policy owner about them.	
Method input attributes		
Name	***Type***	***Description***
conflict_resolution_enabled	boolean	Flag which indicates whether the request should be evaluated with or without conflict resolution.
access_request	String	An XML encoded XACML 3.0 access request.
Method output attributes		
Name	***Type***	***Description***
	String	An XML encoded XACML 3.0 response context.

7.3 Installation guide

This section provides the information on software and hardware requirements to run your own instance of the AU2EU conflict resolution tool. However, since it is implemented as service, it is recommended to simply consume the version that is currently deployed on our demo server, unless you would like to setup your own complete installation of SAFAX.

7.3.1 Hardware requirements

The AU2EU conflict resolution tool is designed to run on commodity server hardware. Due to SAFAX's component based architecture, the hardware of each component can be scaled individually based on one's requirements. We recommend at least:

- 2 GB RAM
- 1 GB hard drive space.
- 2 Cores

7.3.2 Software requirements

In order to run, you will need to install the following software on your system:

A Java Virtual Machine (JVM), version 6 or later. Depending on your machine's architecture, you will need to choose between a 32-bit or 64-bit JVM. We recommend to use a 64-bit implementation if possible. Installation instructions for different Java Versions can be found here:

- Oracle Java: http://www.oracle.com/technetwork/java/javase/downloads/index.html
- OpenJDK: http://openjdk.java.net/install/

An Apache Tomcat servlet container, version 7. Download and installation instructions can be found here:

- http://tomcat.apache.org/whichversion.html

Add the Jersey 2.x libraries to your Tomcat servlet container. The libraries can be downloaded here:

- https://jersey.java.net/download.html

A MySQL community server 5.5. Installation and download instructions can be found here:

- http://dev.mysql.com/downloads/mysql/

SAFAX

- Installation instructions can be found here: http://security1.win.tue.nl/safax/

7.3.3 Installation process

SAFAX is currently provided as Framework as a Service, which is currently deployed on our demo server. The current demo deployment requires credentials to use which are available on request (see http://security1.win.tue.nl/safax/). Developer documentation, user manual and installation instructions can also be found on this web site.

8 Authentication/Authorization translation component

This section describes the detailed architecture of the Authentication-Authorization Mediator (Auth^{2}Mediator) component. We start with a short recapitulation of the overall architecture of the component (cf. [6]). After that we list the concrete requirements derived from that, followed by a discussion about the internal design of the component. At the end of the section we described the RESTful interfaces through which the functions of the component can be used.

Integrating the authorization and authentication system (i.e. XACML and ABC4Trust) will be achieved with this mediator component, which will create a bridging between the two separate systems with providing a mapping between the two types of policies and being able to compose "requests" digestible by either of the systems (i.e. XACML requests and ABC4Trust presentation policies).

8.1 Requirements

After analyzing the use cases and the proposed architectural solution for the integration of the two systems we have identified the following requirements:

Functional requirements

- There has to be a canonicalization of predicates identified and extracted by the enhanced XACML system.
- A proper mapping has to be kept at all times *from* all the encountered predicates in the policies in the authorization system *to* references generated by the Auth^{2}Mediator.
- A proper mapping has to be kept at all times *from* all the generated (or manually created) policies interpretable by the authentication system *to* references generated by the Auth^{2}Mediator.
- The component must be able to compose proper ABC4Trust presentation policy alternatives based on a disjunctive normal form (DNF) of references. These references must come from former calls to the component and the policy composition should be unequivocal.
- The component must be able to compose proper XACML requests based on the response from the authentication system (i.e. a presentation token description) and the DNF which was the basis of the authentication flow. This XACML request must contain all the attributes values (either "True" for predicates or the attribute value itself for disclosed attributes) which were provided by the user through the authentication system.

Non-functional requirements

- The component should have clear interfaces, with clearly separated concerns.
- The component should be replaceable with any other implementation which fully adheres to the same interface contract.
- The mappings should be kept in a fashion which provides consistency (e.g. RDBMS), but not necessarily in one which provides persistency (i.e. in-memory key-value stores are acceptable).

- Other than keeping the mappings for references the component should be stateless, i.e. the parameters to the calls towards this component should contain every information which is needed to render a proper response.

8.2 Design

As introduced in [6] this component will serve multiple purposes besides translating between the authentication and authorization systems.

In Figure 18 we schematically show what internal components are there in the mediator component and how they interact with each other. There is one entry point towards the component – that is the REST API which is defined in the following section. This REST API dispatches the different calls to the corresponding internal components based on the call itself. The internal components are:

- The Reference mappings – is an internal lookup table for storing the predicate to reference and the ABC policy to reference mappings, as described in [6].

- The Predicate interpreter – this internal component interprets the given predicates, namely does the canonicalization and then it compares it against the already existing mappings in the reference mappings. If there is no mapping, it creates one and stores it accordingly.

- The Authentication policy composer – is the internal component which from a disjunctive normal form of references creates a proper ABC presentation policy alternatives with the help of the reference mappings and the DNF interpreter.

- The XACML request composer – this internal component is responsible for creating a proper XACML request based on the results of a successful authentication with the help of the reference mappings and the DNF interpreter.

- The DNF interpreter – is a helper internal component for handling the DNF schema according to the common specification with the authorization system.

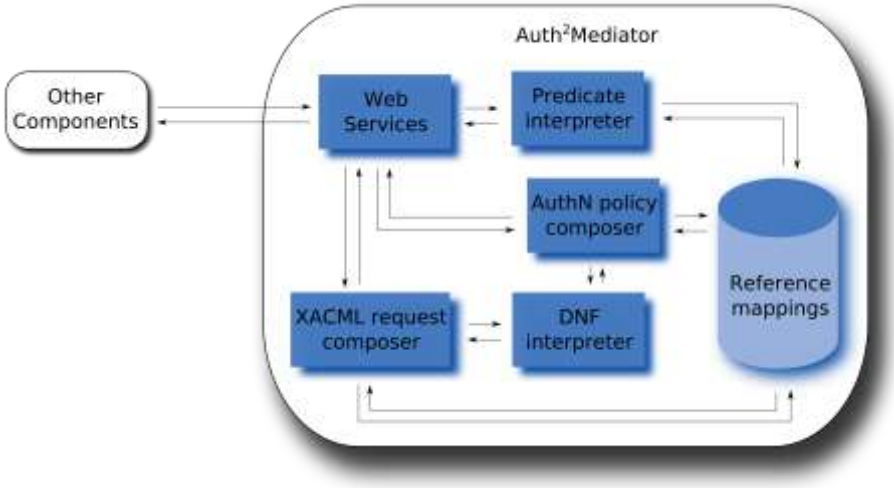

Figure 18 Architecture of the Auth²Mediator component

8.3 REST API

The REST API of the Auth^{2}Mediator component will be able to communicate in XML and JSON as these are the two de-facto standards for transferring object structures in RESTful systems.

It should be noted that these are only wire-formats and the component (and other components throughout the system) should work on the represented object structures and not on the serialized representations. Thus our component is dependent on object structures generated from the XACML XML schema, the ABC4Trust XML schema and a schema describing the structure of the disjunctive normal forms created from the XACML policies containing the references given by this component.

The REST endpoint for looking up an existing or creating a new reference based on a predicate is:

Method Name	*POST/reference*	
Method Definition/Description	Gives back a reference for a given predicate over an attribute.	
Method input attributes		
Name	***Type***	***Description***
Predicate	Object in the payload	An XACML function over values (e.g. "x greater than y") according to the XACML XML schema.
Method output attributes		
Name	***Type***	***Description***
Reference	String	A reference based on the given predicate, either a newly created one or an already existing one.
Dependency with other components	No other than listed in the general description of the REST API.	

The REST endpoint for creating an authentication policy based on a DNF of references created from an authorization policy is:

Method Name	*POST/policy*	
Method Definition/Description	Creates an ABC policy from a given DNF.	
Method input attributes		
Name	***Type***	***Description***
DNF	A DNF object structure in the payload.	The DNF of references from which the ABC policy should be created.
Method output attributes		
Name	***Type***	***Description***

ABC Policy	PresentationPolicyAlternatives	The ABC policy based on the given DNF, according to the ABC4Trust XML schema.
Dependency with other components	The DNF should adhere a common schema between this component and the component which creates it.	

The REST endpoint for rendering an XACML request based on the result coming from the authentication system and the DNF from the original access request call:

Method Name	*attributes*	
Method Definition/Description	Creates an XACML request with attributes based on the results of the authentication component and the corresponding original request.	
Method input attributes		
Name	***Type***	***Description***
Authentication output	PresentationTokenDescription object structure in the payload.	The list of attribute names and values in the validated authentication request. This contains additional information about to which minterm – from the DNF – the attributes are provided to.
DNF	A DNF object structure in the payload.	The DNF of references which was derived from the original request.
Method output attributes		
Name	***Type***	***Description***
XACML request	Request	An XACML request based on the given data.
Dependency with other components	The DNF should adhere a common schema between this component and the component which creates it.	

9 XACML access control framework

The XACML access control framework will be referred to as "Authorization Server". The Authorization Server provides a PDP API: provides an API for getting authorization decisions computed by a XACML-compliant access control engine.

Thales Authorization Server supports the full core XACML 3.0 language; therefore it allows enforcement of generic and complex access control policies.

9.1 Policy Decision Point

The PDP API returns an authorization decision based on the currently enforced policy, access control attributes provided in the request and possibly other attributes resolved by the PDP itself. The Authorization decision is typically Permit, Deny or Not applicable. The PDP is able to resolve extra attributes not provided directly in the request, such as the current time environment attribute or attributes from a presentation token in the request.

9.1.1 REST API

The PDP provides an HTTP RESTful API for requesting authorization decisions. The HTTP request must be formatted as follows:

- Method: POST
- Path: /domains/{domainId}/pdp
- Headers:
  - Content-Type: application/xml
  - Accept: application/xml
- Body: XACML Request as defined in the XACML schema.
- The HTTP response body is a XACML Response as defined in the XACML schema.

9.1.2 Installation guide

The authorization server run on Java, so it is prerequisite to have java running on the server. For compatibility reasons, it is highly recommended to use Sun java instead of the Open Java that is now default for some Linux distributions.

To run the Policy decision Point, you also need a Tomcat Server to deploy the AuthZForce-REST-[VERSION].war (Tomcat 6/0 was our testing version but tomcat 7 can be used too).

To install the software, simply copy the AuthZForce-REST-[VERSION].war in tomcat's webapps folder:

```
[root@authzforce ~]#cp AuthzForce-[VERSION].war /opt/apache-tomcat-6.0.35/webapps/
```

In this version, the default configuration folder is in: /opt/AuthZForce. You need to create it.

```
[root@authzforce ~]# mkdir /etc/AuthZForce

[root@authzforce ~]# mkdir /etc/AuthZForce/conf

[root@authzforce ~]# mkdir /etc/AuthZForce/logs

[root@authzforce ~]# mkdir /etc/AuthZForce/policies
```

Directory	Description
/etc/AuthZForce/conf	Configuration files - log4j.properties→ PDP log4j configuration file - config.xml → PDP configuration file
/etc/AuthZForce/policies	Contains an example policy
/etc/AuthZForce/logs	Logs files - pdp.log → PDP system logs - pdp-audit.log → Authorization decision audit logs

Copy the default configuration file in this directory:

```
[root@authzforce ~]# cp /opt/apache-tomcat-6.0.35/webapps/AuthzForce/WEB-INF/classes/config.xml /etc/AuthZforce/conf/

[root@authzforce ~]# cp /opt/apache-tomcat-6.0.35/webapps/AuthzForce/WEB-INF/classes/log4j.properties /etc/AuthZForce/conf/

[root@authzforce ~]# cp /opt/apache-tomcat-6.0.35/webapps/AuthzForce/WEB-INF/classes/ AuthZForce-Policy.xml /etc/AuthZForce/policies/
```

9.1.3 Installation Checking

1. Start the server by running this command:

```
[root@ authzforce ~]# /opt/apache-tomcat-6.0.35/bin/catalina.sh start
```

2. Test that the PDP and the REST interface is running by doing a GET request to the root (e.g. http://@IP:8080/AuthZForce-REST-[VERSION]/service)

The Server should respond by a 200 status code and display "It Works !"

9.1.4 Example

1 - To test the REST API with AuthZForce you need to get an REST Client like this one for Firefox https://addons.mozilla.org/en-US/firefox/addon/restclient/

2 - Prepare an XACML request fitting your Policy and paste it into your body's request, for us it looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<RequestType xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" ReturnPolicyIdList="false" CombinedDecision="false">
  <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id" IncludeInResult="true">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">bob</AttributeValue>
    </Attribute>
  </Attributes>
  <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" IncludeInResult="true">
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">medicalReccord#81325</AttributeValue>
    </Attribute>
```

```
  </Attributes>
<Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" IncludeInResult="true">
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">medicalReccord#75903</AttributeValue>
    </Attribute>
  </Attributes>
  <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" IncludeInResult="true">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Read</AttributeValue>
    </Attribute>
  </Attributes>
  <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" IncludeInResult="true">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Write</AttributeValue>
    </Attribute>
  </Attributes>
</RequestType>
```

3 - Set the Headers' request with:

```
Content-type : application/xml
```

4 - Send it using a POST request to your PDP's interface (e.g. http://@IP:8080/AuthZForce-REST-[VERSION]/service).

The response should look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<ResponseType xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
  <Result>
    <Decision>Deny</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok" />
      <StatusDetail />
    </Status>
    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
      <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" IncludeInResult="true">
        <AttributeValue                                                                    Da-
taType="http://www.w3.org/2001/XMLSchema#string">medicalReccord#81325</AttributeValue>
      </Attribute>
    </Attributes>
  </Result>
  <Result>
    <Decision>Deny</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok" />
      <StatusDetail />
    </Status>
    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
      <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" IncludeInResult="true">
        <AttributeValue                                                                    Da-
taType="http://www.w3.org/2001/XMLSchema#string">medicalReccord#75903</AttributeValue>
      </Attribute>
    </Attributes>
```

```
  </Result>
  <Result>
    <Decision>Deny</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok" />
      <StatusDetail />
    </Status>
    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
      <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" IncludeInResult="true">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">medicalReccord#81325</AttributeValue>
      </Attribute>
    </Attributes>
  </Result>
  <Result>
    <Decision>Deny</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok" />
      <StatusDetail />
    </Status>
    <Obligations>
      <Obligation ObligationId="urn:oasis:names:tc:xacml:2.0:conformance-test:IIIA014:policyset:obligation-3">
        <AttributeAssignment AttributeId="urn:oasis:names:tc:xacml:2.0:conformance-test:IIIA014:policyset:assignment2"
DataType="http://www.w3.org/2001/XMLSchema#string">assignment2</AttributeAssignment>
      </Obligation>
    </Obligations>
    <AssociatedAdvice />
    <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
      <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" IncludeInResult="true">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">medicalReccord#75903</AttributeValue>
      </Attribute>
    </Attributes>
  </Result>
</ResponseType>
```

10 Mobile authorization platform (NEC)

eHealth and Ambient Assisted Living (AAL) are attracting considerable attention nowadays. In mobile AAL scenarios, health-service-providers wish to integrate their clients' or users' mobile devices in eHealth operations (e.g., secure patient-data provision or provable event/access logging). This clearly raises serious security concerns since the mobile device in question is not under the control of the health-service-provider and is vulnerable to a wide range of security threats. In this section, we address this problem and propose a solution that enhances the security in the eHealth and AAL scenario without compromising the usability, and flexibility of the system. In this deliverable, we focus only on authorization.

Even though for a proper functionality the solution requires a server-backend with attached Mobile Device Management (MDM) infrastructure, in this deliverable we only introduce the mobile part, represented by the flexible mechanism of embedding Policy Enforcement Points (PEP) in combination with a Policy Decision Engine (PDE) inside the mobile device. In future project deliverables, the MDM backend that also is able to push local policies to the mobile devices will be introduced.

10.1 Requirements

Since a proper authorization also in mobile environments is a key security aspect of such systems, our focus is on providing a minimum set of functionalities to fulfill given requirements. These requirements include also:

- Possibility to run the mobile device offline (since network coverage is not provided at all potential locations).
- Flexible design: Policies and Policy Enforcement Points need to be as flexible as possible, since foreseeing all potential use-cases is difficult.
- Compatibility with external running services (e.g., to extend the authorization request to communication entities outside of the device), requiring an online connection.

10.2 Design

The main idea of this approach is to combine an authorization decision engine on mobile devices with additional decision engines running inside a server backend infrastructure.

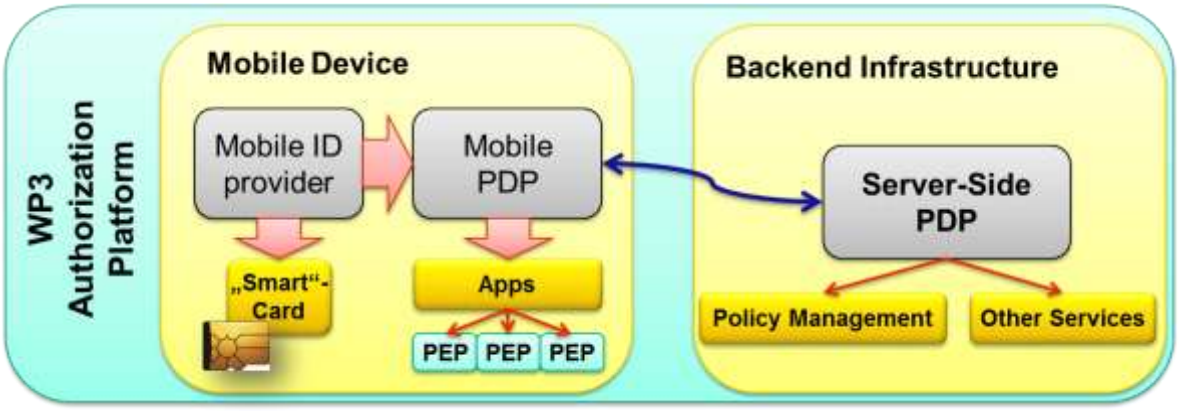

Figure 19: General Mobile Authorization Architecture

The resulting solution gives the possibility to either have local-only policy decisions or remote policy decisions that are enforced on the mobile device.

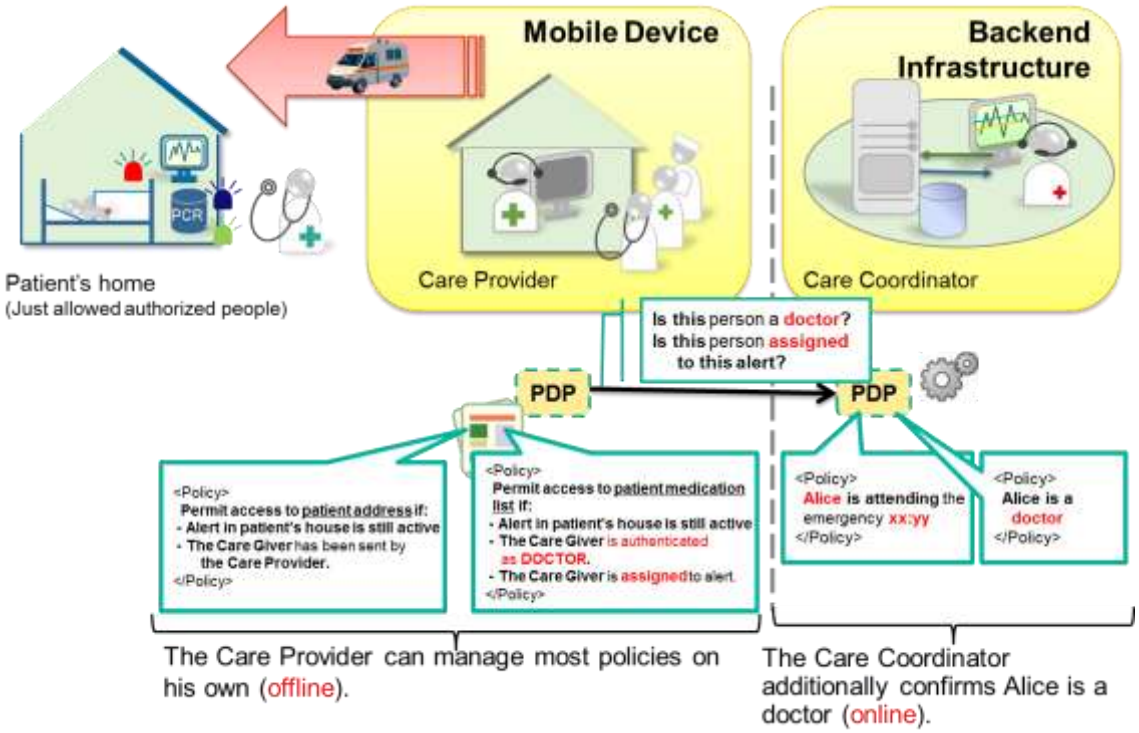
Patient's home
(Just allowed authorized people)
PCR
Mobile Device
Care Provider
Backend Infrastructure
Care Coordinator
Is this person a doctor?
Is this person assigned to this alert?
PDP
PDP
<Policy>
Permit access to patient address if:
- Alert in patient's house is still active
- The Care Giver has been sent by the Care Provider.
</Policy>
<Policy>
Permit access to patient medication list if:
- Alert in patient's house is still active
- The Care Giver is authenticated as DOCTOR.
- The Care Giver is assigned to alert.
</Policy>
<Policy>
Alice is attending the emergency xx:yy
</Policy>
<Policy>
Alice is a doctor
</Policy>
The Care Provider can manage most policies on his own (offline).
The Care Coordinator additionally confirms Alice is a doctor (online).

Figure 20: Online / Offline Policy Decisions

As Figure 20 illustrates, some decisions require online capabilities, others do not require a permanent online connection. This gives the system the possibility to remain functional, even if no Internet connection is available (e.g., a patient is living in an area without proper network coverage).

In case of an available online connection, the local PDP acts as PEP+PIP and creates proper requests to the connected online PDP. Figure 21 further explains how both PDPs are communicating with each other.

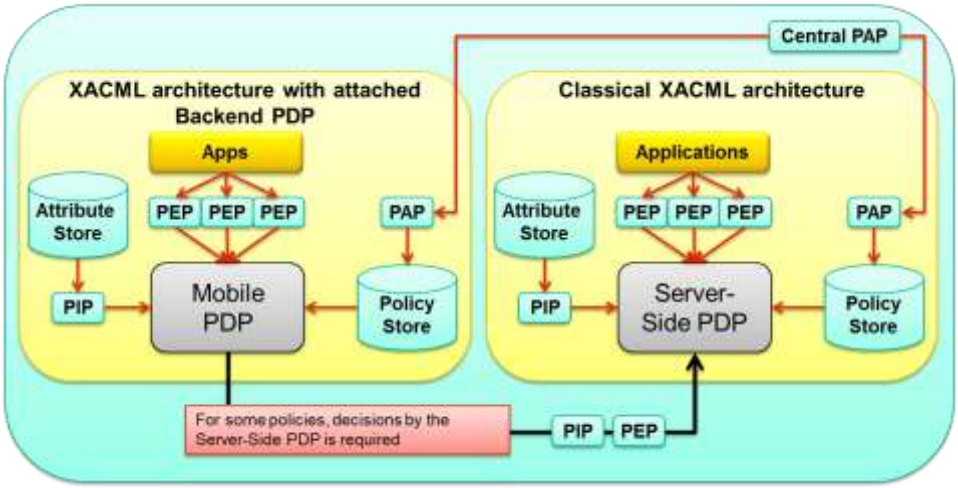

Figure 21: Mobile Device PDP acting as PEP+PIP

10.3 Interfaces

Communication between local and remote PDP is implemented with common XACML requests. No special interfaces had to be implemented, thus an uncomplicated combination and integration of all partner contributions can be achieved.

10.4 Installation guide

10.4.1 Hardware requirements

According to latest development status, the suggested hardware should be equipped with at least 1.5GB or 2GB of runtime memory and a fast CPU similar to the specifications of the Samsung Galaxy S3, which also has been the development phone during project lifetime. Each policy has to be embedded into each involved application, requiring some computation time, depending on the size and complexity of the application. After modification, all applications can be executed without further user interaction. Policies will be automatically properly enforced.

In addition to the mobile Android device, a Smartcard (SC) has to be delivered to each user. Currently, each SC is based on a GlobalPlattform JavaCard 2.x. Each SC is unique per user and offers the possibility to represent the true secure hardware anchor that is needed to provide minimum security requirements on untrusted mobile devices.

10.4.2 Software requirements

Main software requirement is that mobile devices need to be equipped with Android OS running at least version 4.x. Running Android KitKat with SELinux running in enforcement mode (4.4.x) has not been tested yet, thus the current prototype is proven to support at least up to Android OS version 4.3.

Our proposed solution does not require modifications to the underlying operating system of the device and enables IT officials or health-service-providers to remotely manage their desired security and or authorization policies.

10.4.3 Installation process

This solution is based on a usual Android APK file, which has to be installed by the user manually. According to the used Android OS, installation of unsigned APKs has to be enabled manually in the

Android settings. After installation and setting up a few initial settings, such as personal Email, PIN-code for communication with the attached Smartcard and the automatic download of initial policies, installation process is finished.

10.4.4 Troubleshooting

Since our described software is still in a prototypical phase, every potential support incident has to be handled case by case. Future product-ready releases are in discussion.

11 Conclusion

This deliverable describes the first version of the eAuthorization implementation framework. In the following, we summarize the status of each component of the platform and future work.

A prototypical implementation of the ontology-based conversion service was provided as well as the base vocabulary which contains some draft alignments from our use cases to the ontology. For the second period we plan to integrate and test our ontology based conversion service with the AU2EU framework and the use cases. If necessary, we will adopt the implementation as well as the documentation. Furthermore, we will consult the domain experts of the AU2EU use cases to provide their domain vocabulary so that we can finalize the mappings to the ontology.

A first implementation version of the policy validation tool is available. This version is able to transform a policy written in XACML 3.0 into a logic program written in ASP syntax, more especially the Ansprolog syntax. The choice of the true facts of this logic program corresponds to the requests that can be addressed to the PDP. By using "constraint rules" of the program, one can test the responses of the PDP in front of this set of requests, ie, identifying globally the requests producing an expected answer. The approach cannot cover all the syntax of XACML3.0 in terms of calculus. The current version deals only with simple equality conditions in the "XACML targets and conditions". In the second period, we intend to experiment more sophisticated solvers that combine ASP with other logics to improve these limitations and to build test plans for XACML3 policies based on this tool.

The architecture for an implementation of a dynamic conflict resolution solution in a multi-organizational context was proposed. This implementation serves for demonstrating our solution and is based on the SAFAX authorization framework. In the second project phase we intend to test our conflict resolution service with the AU2EU use cases. We will provide meaningful examples and adapt the implementation to be compliant to the XACML standard.

One of the main results of the first period is the design of an integrated eAuthorization and eAuthentication architecture. Such integrated architecture requires augmenting the PDP to support new functionalities, such as mapping of authN and authZ policies and dynamic derivation of attributes representing authentication policies. An authentication Mediator is also needed to be integrated with the XACML asset in order to translate Authorization policies (standard XACML policy) to authentication policies. One of the main objectives of the 2nd period is to implement these extensions.

The new mobile authorization platform for AU2EU project offers a flexible way of policy management for any application on Android OS based mobile devices (e.g., Smartphones or Tablets). Without Application-developer interaction, predefined Policy Enforcement Points (PEPs) and according policies can be embedded to each used application. Embedding happens in a user-transparent way, giving the user a clear overview of what is currently happening on his mobile device, so that a necessary level of trust between user, device and server-side can be guaranteed. From an enterprise perspective, before untrusted devices now can be handled with more trust and embedded to enterprise related use-cases, such as granting access to patient-privacy-critical information. By combining the untrusted user-device with trusted entities (here: smartcards with NFC or microSD based communication interfaces), a reasonable level of a trade-off between security and usability can be established. Having a Policy Decision Point (PDP) not only on servers running with permanent network connec-

tion but also on the mobile device itself, gives the mobile devices the possibility to decide based on their own internal policy-database and offers offline decisions, in contrast to online decisions if the device is connected to the servers.

The current status of the mobile authorization platform is that the design has been finished and first prototypes are present. Together with the German Red Cross, NEC now enters the next phase of connecting everything to the real use-case, i.e., the efficient care coordination use-case and the customer registration use-case. In addition, by connecting the prototypes to the use-cases, the connection to the server machines with additional authorization-logic has to be done. According to the project plan, this implementation and roll-out will be focused in the 2nd project period.

Bibliography

[1] W. Burr, D. Dodson, E. Newton, R. Perlner, W. Polk, S. Gupta and E. Nabbus, *Electronic Authentication Guideline, NIST Special Publication 800-63-2.*

[2] D. Trivellato, N. Zannone, M. Glaundrup, J. Skowronek and S. Etalle, "A Semantic Security Framework for Systems of Systems," *Int. J. Cooperative Inf. Syst. 22(1),* 2013.

[3] K. Bohm, S. Etalle, J. d. Hartog, C. Hutter, S. Trabelsi, D. Trivellato and N. Zannone, "A Flexible Architecture for Privacy-Aware Trust Management," *Journal of Theoretical and Applied Electronic Commerce Research,* 2010.

[4] S. W. Group, "SPARQL 1.1 W3C Recommendation," 21 03 2013. [Online]. Available: http://www.w3.org/TR/rdf-sparql-query/. [Accessed 19 11 2014].

[5] "eXtensible Access Control Markup Language (XACML) Version 3.0," OASIS, 22 01 2013. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html. [Accessed 9 11 2014].

[6] A. Consortium, D3.2.1 - Architecture for Integrating Authorization with eAuthentication, 2014.

A. Appendix

a. Policy1

```
<?xml version="1.0"?>
<Policy     xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"     Poli-
cyId="P0Policy"      RuleCombiningAlgId="urn:oasis:names:tc:xacml:3.0:rule-
combining-algorithm:deny-overrides" Version="1.0">
  <Target/>
  <Rule Effect="Permit" RuleId="manager-ac">
    <Target>
      <AnyOf>
        <AllOf>
          <Match     MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">operator</AttributeValue
>
            <AttributeDesignator                                          At-
tributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"          Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </Match>
        <Match       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">manager</AttributeValue>
            <AttributeDesignator                                          At-
tributeId="urn:oasis:names:tc:xacml:2.0:subject:role"              Catego-
ry="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </Match>
        </AllOf>
      </AnyOf>
    </Target>
  </Rule>
  <Rule Effect="Permit" RuleId="engineer-ac">
    <Target>
      <AnyOf>
        <AllOf>
          <Match     MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">admin</AttributeValue>
            <AttributeDesignator                                          At-
tributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"          Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </Match>
        <Match       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">engineer</AttributeValue
>
            <AttributeDesignator                                          At-
tributeId="urn:oasis:names:tc:xacml:2.0:subject:role"              Catego-
ry="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </Match>
        </AllOf>
```

```
      </AnyOf>
    </Target>
  </Rule>
  <Rule Effect="Permit" RuleId="tech-ac">
    <Target>
      <AnyOf>
        <AllOf>
          <Match      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">admin</AttributeValue>
            <AttributeDesignator                                          At-
tributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"          Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </Match>
        <Match       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">technician</AttributeVal
ue>
            <AttributeDesignator                                          At-
tributeId="urn:oasis:names:tc:xacml:2.0:subject:role"              Catego-
ry="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </Match>
        <Match     MatchId="urn:oasis:names:tc:xacml:1.0:function:time-less-
than-or-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#time">08:00:00</AttributeValue>
          <AttributeDesignator                                            At-
tributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"   Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true"/>
        </Match>
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:time-greater-
than-or-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#time">22:00:00</AttributeValue>
          <AttributeDesignator                                            At-
tributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"   Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true"/>
        </Match>
        </AllOf>
      </AnyOf>
    </Target>
    <Condition>
      <Apply      FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
      <Apply  FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-
and-only">
        <AttributeDesignator                                          Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"          At-
tributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-location"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
      </Apply>
      <Apply  FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-
and-only">
```

```
        <AttributeDesignator                                    Catego-
ry="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"       At-
tributeId="urn:oasis:names:tc:xacml:1.0:subject:user-location"
DataType="http://www.w3.org/2001/XMLSchema#string"   MustBePresent="true"/>

      </Apply>
      </Apply>
    </Condition>
  </Rule>
  <Rule Effect="Permit" RuleId="operator-ac">
    <Target>
      <AnyOf>
        <AllOf>
          <Match     MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">operator</AttributeValue
>
            <AttributeDesignator                                    At-
tributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"          Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </Match>
        <Match       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">operator</AttributeValue
>
            <AttributeDesignator                                    At-
tributeId="urn:oasis:names:tc:xacml:2.0:subject:role"              Catego-
ry="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </Match>
        <Match    MatchId="urn:oasis:names:tc:xacml:1.0:function:time-less-
than-or-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#time">08:00:00</AttributeValue>
          <AttributeDesignator                                    At-
tributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"   Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true"/>
        </Match>
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:time-greater-
than-or-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#time">22:00:00</AttributeValue>
          <AttributeDesignator                                    At-
tributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"   Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true"/>
        </Match>
        </AllOf>
      </AnyOf>
    </Target>
    <Condition>
      <Apply       FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
      <Apply  FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-
and-only">
```

```
        <AttributeDesignator                                    Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"          At-
tributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-location"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
      </Apply>
      <Apply  FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-
and-only">
        <AttributeDesignator                                    Catego-
ry="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"      At-
tributeId="urn:oasis:names:tc:xacml:1.0:subject:user-location"
DataType="http://www.w3.org/2001/XMLSchema#string"   MustBePresent="true"/>

      </Apply>
      </Apply>
    </Condition>
  </Rule>
  <Rule Effect="Permit" RuleId="serviceDesk-tech">
    <Target>
      <AnyOf>
        <AllOf>
          <Match     MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">admin</AttributeValue>
            <AttributeDesignator                                    At-
tributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"          Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </Match>
        <Match       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">technician</AttributeVal
ue>
            <AttributeDesignator                                    At-
tributeId="urn:oasis:names:tc:xacml:2.0:subject:role"              Catego-
ry="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </Match>
        <Match    MatchId="urn:oasis:names:tc:xacml:1.0:function:time-less-
than-or-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#time">08:00:00</AttributeValue>
          <AttributeDesignator                                    At-
tributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"   Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true"/>
        </Match>
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:time-greater-
than-or-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#time">22:00:00</AttributeValue>
          <AttributeDesignator                                    At-
tributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"   Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true"/>
        </Match>
        </AllOf>
      </AnyOf>
```

```
    </Target>
    <Condition>
      <Apply      FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-
equal">
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#boolean">true</AttributeValue>
        <Apply    FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-
one-and-only">
        <AttributeDesignator                                          At-
tributeId="urn:oasis:names:tc:xacml:2.0:subject:servicedesk"        Catego-
ry="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
DataType="http://www.w3.org/2001/XMLSchema#boolean" MustBePresent="true"/>
        </Apply>
      </Apply>
    </Condition>
  </Rule>
  <Rule Effect="Permit" RuleId="serviceDesk-operator">
    <Target>
      <AnyOf>
        <AllOf>
          <Match      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">operator</AttributeValue
>
            <AttributeDesignator                                      At-
tributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"           Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </Match>
        <Match        MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">operator</AttributeValue
>
            <AttributeDesignator                                      At-
tributeId="urn:oasis:names:tc:xacml:2.0:subject:role"               Catego-
ry="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </Match>
        <Match     MatchId="urn:oasis:names:tc:xacml:1.0:function:time-less-
than-or-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#time">08:00:00</AttributeValue>
          <AttributeDesignator                                        At-
tributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"   Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true"/>
        </Match>
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:time-greater-
than-or-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#time">22:00:00</AttributeValue>
          <AttributeDesignator                                        At-
tributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"   Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true"/>
        </Match>
        </AllOf>
```

```
      </AnyOf>
    </Target>
    <Condition>
      <Apply      FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-
equal">
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#boolean">true</AttributeValue>
        <Apply    FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-
one-and-only">
        <AttributeDesignator                                          At-
tributeId="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
DataType="http://www.w3.org/2001/XMLSchema#boolean" MustBePresent="true"/>
        </Apply>
      </Apply>
    </Condition>
  </Rule>
  <Rule Effect="Permit" RuleId="incident-operator">
    <Target>
      <AnyOf>
        <AllOf>
          <Match      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">operator</AttributeValue
>
            <AttributeDesignator                                      At-
tributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"          Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </Match>
        <Match        MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">operator</AttributeValue
>
            <AttributeDesignator                                      At-
tributeId="urn:oasis:names:tc:xacml:2.0:subject:role"             Catego-
ry="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </Match>
        <Match     MatchId="urn:oasis:names:tc:xacml:1.0:function:time-less-
than-or-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#time">08:00:00</AttributeValue>
          <AttributeDesignator                                        At-
tributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"   Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true"/>
        </Match>
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:time-greater-
than-or-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#time">22:00:00</AttributeValue>
          <AttributeDesignator                                        At-
tributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"   Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true"/>
        </Match>
```

```
        </AllOf>
      </AnyOf>
    </Target>
    <Condition>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
      <Apply      FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
        <Apply    FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
one-and-only">
          <AttributeDesignator                                    Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"          At-
tributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-location"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </Apply>
        <Apply    FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
one-and-only">
          <AttributeDesignator                                    Catego-
ry="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"      At-
tributeId="urn:oasis:names:tc:xacml:1.0:subject:user-location"
DataType="http://www.w3.org/2001/XMLSchema#string"   MustBePresent="true"/>

        </Apply>
      </Apply>
      <Apply     FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-
equal">
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#boolean">true</AttributeValue>
        <Apply   FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-
one-and-only">
          <AttributeDesignator                                       At-
tributeId="urn:oasis:names:tc:xacml:2.0:subject:incident"          Catego-
ry="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
DataType="http://www.w3.org/2001/XMLSchema#boolean" MustBePresent="true"/>
        </Apply>
      </Apply>
      </Apply>
    </Condition>
  </Rule>
  <Rule Effect="Permit" RuleId="incident-tech">
    <Target>
      <AnyOf>
        <AllOf>
          <Match     MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">admin</AttributeValue>
            <AttributeDesignator                                     At-
tributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"          Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </Match>
        <Match       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">technician</AttributeVal
ue>
            <AttributeDesignator                                     At-
tributeId="urn:oasis:names:tc:xacml:2.0:subject:role"              Catego-
```

```
ry="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
          </Match>
        <Match    MatchId="urn:oasis:names:tc:xacml:1.0:function:time-less-
than-or-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#time">08:00:00</AttributeValue>
          <AttributeDesignator                                          At-
tributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"   Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true"/>
        </Match>
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:time-greater-
than-or-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#time">22:00:00</AttributeValue>
          <AttributeDesignator                                          At-
tributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"   Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
DataType="http://www.w3.org/2001/XMLSchema#time" MustBePresent="true"/>
        </Match>
        </AllOf>
      </AnyOf>
    </Target>
    <Condition>

      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
      <Apply      FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
        <Apply    FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
one-and-only">
          <AttributeDesignator                                  Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"          At-
tributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-location"
DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
        </Apply>
        <Apply    FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-
one-and-only">
          <AttributeDesignator                                  Catego-
ry="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"      At-
tributeId="urn:oasis:names:tc:xacml:1.0:subject:user-location"
DataType="http://www.w3.org/2001/XMLSchema#string"   MustBePresent="true"/>

        </Apply>
      </Apply>
      <Apply     FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-
equal">
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#boolean">true</AttributeValue>
        <Apply   FunctionId="urn:oasis:names:tc:xacml:1.0:function:boolean-
one-and-only">
          <AttributeDesignator                                          At-
tributeId="urn:oasis:names:tc:xacml:2.0:subject:incident"          Catego-
ry="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
DataType="http://www.w3.org/2001/XMLSchema#boolean" MustBePresent="true"/>
        </Apply>
      </Apply>
      </Apply>
    </Condition>
```

```
  </Rule>
</Policy>
```

b. Policy2

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
PolicyId="P3RBlackList"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:3.0:rule-combining-
algorithm:deny-overrides"
Version="1.0">
  <Target />
  <Rule Effect="Deny" RuleId="maq7">
    <Target>
      <AnyOf>
        <AllOf>
          <Match      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">

            <AttributeValue                                                Da-
taType="http://www.w3.org/2001/XMLSchema#string">
            maq8</AttributeValue>
            <AttributeDesignator                                           At-
tributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
            Category="urn:oasis:names:tc:xacml:3.0:attribute-
category:resource"
            DataType="http://www.w3.org/2001/XMLSchema#string"
            MustBePresent="true" />
          </Match>
        </AllOf>
      </AnyOf>
    </Target>
  </Rule>
</Policy>
```

c. Policy3

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
      PolicyId="P4Inactive"
      RuleCombiningAlgId="urn:oasis:names:tc:xacml:3.0:rule-combining-
algorithm:deny-overrides"
      Version="1.0">
  <Target />

  <Rule Effect="Deny" RuleId="bl-maria">
    <Target>
      <AnyOf>
      <AllOf>
        <Match       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Maria
          </AttributeValue>
          <AttributeDesignator                                             At-
tributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"         Catego-
ry="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
DataType="http://www.w3.org/2001/XMLSchema#string"    MustBePresent="true"/>
```

```
        </Match>
        <Match    MatchId="urn:oasis:names:tc:xacml:1.0:function:date-less-
than-or-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#date">2014-09-12
          </AttributeValue>
          <AttributeDesignator                                          At-
tributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"   Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
DataType="http://www.w3.org/2001/XMLSchema#date" MustBePresent="true"/>
        </Match>
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:date-greater-
than-or-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#date">2014-09-21
          </AttributeValue>
          <AttributeDesignator                                          At-
tributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"   Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
DataType="http://www.w3.org/2001/XMLSchema#date" MustBePresent="true"/>
        </Match>
      </AllOf>
      </AnyOf>
    </Target>
  </Rule>

  <Rule Effect="Deny" RuleId="bl-daniel">
    <Target>
      <AnyOf>
      <AllOf>
        <Match       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Daniel
          </AttributeValue>
          <AttributeDesignator                                          At-
tributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"         Catego-
ry="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
DataType="http://www.w3.org/2001/XMLSchema#string"   MustBePresent="true"/>

        </Match>
        <Match    MatchId="urn:oasis:names:tc:xacml:1.0:function:date-less-
than-or-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#date">2014-09-19
          </AttributeValue>
          <AttributeDesignator                                          At-
tributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"   Catego-
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
DataType="http://www.w3.org/2001/XMLSchema#date" MustBePresent="true"/>
        </Match>
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:date-greater-
than-or-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#date">2014-09-28
          </AttributeValue>
          <AttributeDesignator                                          At-
tributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"   Catego-
```

```
ry="urn:oasis:names:tc:xacml:3.0:attribute-category:environment"
DataType="http://www.w3.org/2001/XMLSchema#date" MustBePresent="true"/>
        </Match>
      </AllOf>
      </AnyOf>
    </Target>
  </Rule>

</Policy>
```

d. Policy4

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
PolicyId="P2BlackList"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:3.0:rule-combining-
algorithm:deny-overrides"
Version="1.0">
  <Target />
  <Rule Effect="Permit" RuleId="aw-jordi">
    <Target>
      <AnyOf>
        <AllOf>
          <Match     MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">

            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">
            Jordi</AttributeValue>
            <AttributeDesignator                                    At-
tributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
            Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
subject"
            DataType="http://www.w3.org/2001/XMLSchema#string"
            MustBePresent="true" />
          </Match>
          <Match     MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
equal">

            <AttributeValue                                         Da-
taType="http://www.w3.org/2001/XMLSchema#string">
            maq7</AttributeValue>
            <AttributeDesignator                                    At-
tributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
            Category="urn:oasis:names:tc:xacml:3.0:attribute-
category:resource"
            DataType="http://www.w3.org/2001/XMLSchema#string"
            MustBePresent="true" />
          </Match>
          <Match  MatchId="urn:oasis:names:tc:xacml:1.0:function:date-less-
than-or-equal">

            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#date">
            2014-09-12</AttributeValue>
            <AttributeDesignator                                    At-
tributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"
```

```
            Category="urn:oasis:names:tc:xacml:3.0:attribute-
category:environment"
            DataType="http://www.w3.org/2001/XMLSchema#date"
            MustBePresent="true" />
          </Match>
          <Match       MatchId="urn:oasis:names:tc:xacml:1.0:function:date-
greater-than-or-equal">

            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#date">
            2014-09-14</AttributeValue>
            <AttributeDesignator                                          At-
tributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"
            Category="urn:oasis:names:tc:xacml:3.0:attribute-
category:environment"
            DataType="http://www.w3.org/2001/XMLSchema#date"
            MustBePresent="true" />
          </Match>
        </AllOf>
      </AnyOf>
    </Target>
  </Rule>
      </Policy>
```